GNU Aspell 0.60.4

Kevin Atkinson (kevina@gnu.org)

This is the user's manual for Aspell

Copyright © 2000–2005 Kevin Atkinson.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

Table of Contents

1 Introduction 1

1.1 Comparison to other spell checker engines 1
1.1.1 Comparison to Ispell 1
1.1.1.1 Features that only Aspell has 1
1.1.1.2 Things that, currently, only Ispell has 2

2 Support 3

3 Basic Usage 4

3.1 Spellchecking Individual Files 4
3.2 Using Aspell as a Replacement for Ispell 5
3.2.1 As a Drop In Replacement 5
3.2.2 Differences From Ispell 5
3.2.2.1 Functionally of the Ispell Compatibility Script 6
3.2.2.2 Recognized Options 6
3.2.2.3 Check Mode Compatibility 6
3.2.2.4 Pipe Mode Compatibility 6
3.2.2.5 Other Differences 6
3.2.2.6 Missing Functionally 7
3.3 Using Aspell with other Applications 7
3.3.1 With Emacs and XEmacs 7
3.3.2 With LyX 7
3.3.3 With VIM 7
3.3.4 With Pine 7

4 Customizing Aspell 9

4.1 Specifying Options 9
4.1.1 At the Command Line 9
4.1.1.1 Value options 9
4.1.1.2 Boolean options 9
4.1.1.3 List options 10
4.1.2 Via a Configuration File 10
4.1.2.1 Value options 10
4.1.2.2 Boolean options 10
4.1.2.3 List options 11
4.1.3 Setting Options via an Environment Variable 11
4.2 The Options 11
4.2.1 Dictionary Options 11
4.2.2 Encoding Options 12
4.2.3 Checker Options 12
4.2.4 Filter Options 13
4.2.4.1 email 13

4.2.4.2 html . . . 13
4.2.4.3 sgml . . . 14
4.2.4.4 tex/latex . . . 14
4.2.4.5 texinfo . . . 14
4.2.4.6 context . . . 14
4.2.5 Run-together Word Options . . . 14
4.2.6 Miscellaneous Options . . . 15
4.2.7 Aspell Utility Options . . . 15
4.3 Dumping Configuration Values . . . 16
4.4 Notes on Various Options . . . 16
4.4.1 Notes on Various Filters and Filter Modes . . . 16
4.4.1.1 None Filter Mode . . . 16
4.4.1.2 URL Filter . . . 16
4.4.1.3 Email Filter . . . 16
4.4.1.4 SGML Filter . . . 16
4.4.1.5 HTML Filter . . . 17
4.4.1.6 TEX/LaTeX Filter . . . 17
4.4.1.7 Texinfo Filter . . . 18
4.4.1.8 Nroff Filter . . . 18
4.4.1.9 Context Filter . . . 19
4.4.1.10 Ccpp Filter Mode . . . 19
4.4.2 Notes on the Prefix Option . . . 19
4.4.3 Notes on Typo-Analysis and the Keyboard Definition File . . . 19
4.4.4 Notes on the Different Suggestion Modes . . . 20

5 Working With Dictionaries . . . 21

5.1 Using `aspell-import` . . . 21
5.2 How Aspell Selects an Appropriate Dictionary . . . 21
5.3 Listing Available Dictionaries . . . 21
5.4 Dumping the Contents of the Word List . . . 22
5.5 Creating an Individual Word List . . . 22
5.5.1 Format of the Replacement Word List . . . 23
5.6 Working With Affix Info in Word Lists . . . 23
5.6.1 The Munch Command . . . 23
5.6.2 The Expand Command . . . 23
5.6.3 The Munch-list Command . . . 23
5.7 Format of the Personal and Replacement Dictionaries . . . 24
5.7.1 Format of the Personal Dictionary . . . 24
5.7.2 Format of the Personal Replacement Dictionary . . . 24
5.8 Using Multi Dictionaries . . . 25
5.9 Dictionary Naming . . . 25
5.10 AWLI files . . . 25

6 Writing programs to use Aspell 26
6.1 Through the C API 26
6.1.1 Usage 26
6.1.2 API Reference 28
6.1.3 Examples 28
6.1.4 Notes About Thread Safety 28
6.2 Through A Pipe 28
6.2.1 Format of the Data Stream 28
6.3 Notes on Storing Replacement Pairs 30

7 Adding Support For Other Languages 31
7.1 The Language Data File 31
7.2 Compiling the Word List 33
7.3 Phonetic Code 33
7.3.1 Syntax of the transformation array 33
7.3.2 How do I start finally? 35
7.3.2.1 Things that come in handy 36
7.3.2.2 What the phonetic code should do 36
7.4 The Simple Soundslike 37
7.5 Replacement Tables 38
7.6 Affix Compression 38
7.6.1 Format of the Affix File 39
7.6.2 When Compared With Ispell 40
7.6.3 Specifying Affix Flags 40
7.7 Controlling the Behavior of Run-together Words 40
7.8 Creating A New Character Set 41
7.9 Format of the Character Set Data File 41
7.10 Creating An Official Dictionary Package 41

Appendix A Implementation Notes 42
A.1 Aspell Suggestion Strategy 42
A.2 Notes on 8-bit Characters 42

Appendix B Languages Which Aspell can Support 44
B.1 Supported 44
B.1.1 Notes on Latin Languages 48
B.1.2 Syllabic 49
B.1.2.1 The Ethiopic Syllabary 49
B.1.2.2 The Yi Syllabary 50
B.1.2.3 The Ojibwe Syllabary 50
B.2 Unsupported 50
B.2.1 The Thai, Khmer, and Lao Scripts 50
B.2.2 Languages which use Hànzi Characters 50
B.2.3 Japanese 51
B.2.4 Hangul 51
B.3 Languages Written in Multiple Scripts 51

B.4 Notes on Planned Dictionaries 51
B.5 References 53

Appendix C Language Related Issues 54
C.1 Compound Words 54
C.1.1 To Implement 55
C.2 Words With Spaces or Other Symbols in Them 55
C.3 Unicode Normalization 56
C.4 German Sharp S 56
C.5 Context Sensitive Spelling 56

Appendix D To Do 57
D.1 Important Items 57
D.1.1 Things that need to be done 57
D.1.2 Things I would like to get done 57
D.2 Other Items 58
D.3 Notes on Various Items 59
D.3.1 Word skipping by context 59
D.3.2 Hidden Markov Model 59
D.3.3 Email the Personal Dictionary 60

Appendix E Installing 61
E.1 Generic Install Instructions 61
E.2 HTML Manuals and `make clean` 61
E.3 Curses Notes 61
E.3.1 Unicode Support 62
E.4 Loadable Filter Notes 62
E.5 Upgrading from Aspell 0.50 62
E.5.1 Binary Compatibility 63
E.6 Upgrading from Aspell .33/Pspell .12 63
E.7 Upgrading from a Pre-0.50 snapshot 64
E.8 WIN32 Notes 64
E.8.1 Getting the WIN32 version 64
E.8.2 Building the WIN32 version 64
E.8.2.1 Building Aspell using Cygwin 64
E.8.2.2 Building Aspell using MinGW 64
E.8.2.3 Building Aspell without using Cygwin or MSYS 64
E.8.3 (PD)Curses 65
E.8.4 Directories 66
E.8.5 Installer 66
E.8.6 WIN32 consoles 66

Appendix F ChangeLog 67

Appendix G Authors 82

Appendix H Copying 84
H.1 GNU Free Documentation License 84
H.1.1 ADDENDUM: How to use this License for your documents 91
H.2 GNU Lesser General Public License 91
H.2.1 Preamble 91
H.2.2 TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION 93
H.2.3 How to Apply These Terms to Your New Libraries 99

1 Introduction

GNU Aspell is a spell checker designed to eventually replace Ispell. It can either be used as a library or as an independent spell checker. Its main feature is that it does a much better job of suggesting possible replacements for a misspelled word than just about any other spell checker out there for the English language. Unlike Ispell, Aspell can also easily check documents in UTF-8 without having to use a special dictionary. Aspell will also do its best to respect the current locale setting. Other advantages over Ispell include support for using multiple dictionaries at once and intelligently handling personal dictionaries when more than one Aspell process is open at once.

The latest version of Aspell can always be found at `http://aspell.net`

1.1 Comparison to other spell checker engines

	Aspell	Ispell	Netscape 4.0	Microsoft Word 97
Open Source	x	x		
Suggestion Intelligence	88-98	54	55-70?	71
Personal part of Suggestions	x	x	x	
Alternate Dictionaries	x	x	?	?
International Support	x	x	?	?

The Suggestion Intelligence is based on a small test kernel of misspelled/correct word pairs. Go to `http://aspell.net/test` for more info and how you can help contribute to the test kernel. The current scores for Aspell are 88 in *fast* mode, 93 in *normal* mode, and 98 in *bad spellers* mode: for more information about the various suggestion modes Section 4.4.4 [Notes on the Different Suggestion Modes], page 20.

If you have any other information you would like to add to this chart please contact me at `kevina@gnu.org`.

1.1.1 Comparison to Ispell

1.1.1.1 Features that only Aspell has

- Is an actual library that other programs can link to instead of having to use it through a pipe.
- Does a much better job of suggesting possible replacements for a misspelled word than Ispell does or for that matter any other spell checker I have seen. If you know a spell checker that does a better job please let me know.
- Can learn from user's misspellings.
- Can easily check documents in UTF-8 without having to use a special dictionary.
- Has support for using multiple dictionaries at once.
- Is multiprocess intelligent. When a personal dictionary (or replacement list) is saved it will now first update the list against the dictionary on disk in case another process modified it.

- Can share the memory used in the main word list between processes.
- A better, more complete word list for the English language. Word lists are provided for American, British, and Canadian spelling. Special care has been taken to only include one spelling for each word in any particular word list. The word list included in Ispell by contrast only included support for American and British and also tends to included multiple spellings for a word which can mask some spelling errors.

1.1.1.2 Things that, currently, only Ispell has

- Lower memory footprint
- Ability to deal with arbitrary multi-character letters such as old ASCII encodings of accented letters.
- Perhaps better support for spell checking (La)TeX files.

For a detailed description of how Aspell differs from Ispell, See [Differences From Ispell], page 5.

2 Support

Support for Aspell can be found on the Aspell mailing lists. Instructions for joining the various mailing lists (and an archive of them) can be found off the Aspell home page at `http://aspell.net`.

Bug reports should be submitted via the Sourceforge Tracker at `http://sourceforge.net/tracker/?group_id=245` rather than being posted to the mailing lists.

3 Basic Usage

For a quick reference on the Aspell utility use the command `aspell --help`.

3.1 Spellchecking Individual Files

To use Aspell to spellcheck a file use:

```
aspell check [options] filename
```

at the command line where *`filename`* is the file you want to check and *`options`* is any number of optional options. Some of the more useful ones include:

–**mode**=*mode*
: the mode to use when checking files. The available modes are `none`, `url`, `email`, `sgml`, `tex`, `texinfo`, `nroff`, among others. For more information on the various modes see Section 4.4.1 [Notes on Various Filters and Filter Modes], page 16.

–**dont-backup**
: don't create a backup file.

–**sug-mode**=*mode*
: the suggestion mode to use where mode is one of `ultra`, `fast`, `normal`, or `bad-spellers`. For more information on these modes see Section 4.4.4 [Notes on the Different Suggestion Modes], page 20.

–**lang**=*name*/**-l** *name*
: the language the document is written in. The default depends on the current locale.

–**encoding**=*name*
: encoding the document is expected to be in. The default depends on the current locale.

–**master**=*name*/**-d** *name*
: the main dictionary to use.

–**keymapping**=*name*
: the keymapping to use. Either '`aspell`' for the default mapping or '`ispell`' to use the same mapping that the Ispell utility uses.

For more information on the available options, please see Chapter 4 [Customizing Aspell], page 9.

For example to check the file '`foo.txt`':

```
aspell check foo.txt
```

and to check the file '`foo.txt`' using the '`bad-spellers`' suggestion mode and the American English dictionary:

```
aspell check --sug-mode=bad-spellers -d en_US foo.txt
```

If the '`mode`' option is not given then Aspell will use the extension of the file to determine the current mode. If the extension is '`.tex`', then '`tex`' mode will be used, if the extension is '`.html`', '`.htm`', '`.php`', or '`.sgml`' it will check the file in '`sgml`' mode, otherwise it will use '`url`' mode.

For more information on the various modes that can be used, see Section 4.4.1 [Notes on Various Filters and Filter Modes], page 16.

If Aspell was compiled with curses support and the `TERM` environment variable is set to a capable terminal type than Aspell will use a nice full screen interface. Otherwise it will use a simpler "dumb" terminal interface where the misspelled word is surrounded by two '*'. In either case the interface should be self explanatory.

If Aspell is compiled with a version of the curses library that support wide characters than Aspell can also check UTF-8 text. Furthermore, the document will be displayed in the encoding defined by the current locale. This encoding does not necessary have to be the same encoding that the document is in. This means that is is possible to check an 8-bit encoding such as ISO-8859-1 on an UTF-8 terminal. To do so simply set the '`encoding`' option to '`iso-8859-1`'. Furthermore it is also possible to check an UTF-8 document on an 8-bit terminal provided that the document can be successfully converted into that encoding.

3.2 Using Aspell as a Replacement for Ispell

As of GNU Aspell 0.60.1 Aspell should be able to completely replace Ispell for most applications. The Ispell compatibility script should work for most applications which expect Ispell. However their are some differences which you should be aware of.

3.2.1 As a Drop In Replacement

Aspell can be used as a drop in replacement for Ispell for programs that use Ispell through a pipe such as Emacs and LyX. It can also be used with programs that simply call the `ispell` command and expect the original file to be overwritten with the corrected version.

If you do not have Ispell installed on your system and have installed the Ispell compatibly script than you should not need to do anything as most applications will that expect Ispell will work as expected with Aspell via the Ispell compatibility script.

Otherwise, the recommended way to use Aspell as a replacement for Ispell is to change the `ispell` command from within the program being used. If the program uses `ispell` in pipe mode simple change `ispell` to `aspell`. If the program calls the `ispell` command to check the file change `ispell` with `aspell check`.

If that is impossible than the `run-with-aspell` script can be used. This script modifies the patch so that programs see the Ispell compatibility script instead of the actual true `ispell` command. The format of the script is:

```
run-with-aspell command
```

where *command* is the name of the program with any optional arguments.

The old method of mapping Ispell to Aspell is discouraged because it can create compatibility problems with programs that actually require Ispell such as Ispell's own scripts.

3.2.2 Differences From Ispell

Nevertheless, Aspell is not Ispell, nor is it meant to completely emulate the behavior of Ispell. The `aspell` command is not identical to the `ispell` command when not used in "pipe" mode. If an application expects the `ispell` command than the Ispell compatibility script should be used.

3.2.2.1 Functionally of the Ispell Compatibility Script

The Ispell compatibility script provides the following Ispell functionally.

- The ability to check a file when called with out any mode parameters.
- The pipe or -a mode.
- The list of -l mode.
- The version or -v mode. A single line is returned which, while not being identical to the line Ispell returns, is sufficient to fool most programs.
- The munch or -c mode.
- The expand or -e mode.
- The ability to dump the affix file when called with '-D'. However the format of the affix file is different. Furthermore, not all languages have an affix file.

However the Ispell script currently unable to emulate the the '-A' pipe mode. This is different from the normal pipe mode in that the special `&Include_File&` command is recognized.

3.2.2.2 Recognized Options

Aspell, and thus the ispell compatibility script, recognizes most of the options that Ispell uses except for the '-S', '-w' and '-T' options. The Aspell command will simply ignore these options if it sees them.

3.2.2.3 Check Mode Compatibility

The interface used by Aspell when checking individual files is slightly different than Ispell's. In particular the default keymappings are not the same as the ones Ispell uses. However Aspell supports using the Ispell keymappings via the '`keymapping`' option. To use the Ispell keymappings set the '`kepmapping`' option to `ispell`. This can be done on the command line by adding using the command:

```
aspell check --keymapping=ispell ...
```

or with the Ispell compatibility script

```
ispell --keymapping=ispell ...
```

The Ispell keymapping can always be used when the Ispell compatibly script is called by uncommenting the indicated line in the `ispell` script.

3.2.2.4 Pipe Mode Compatibility

The Aspell pipe mode should be identical to the Ispell pipe mode except if the line starts with a '$$' as that will trigger special Aspell only commands or if the line starts with a '~' which is ignored by Aspell.

3.2.2.5 Other Differences

The compiled dictionary format is completely different than Ispell's. Furthermore the format of the language data files' are different than Ispell's affix file. However, all known Ispell dictionaries been converted to Aspell format, except for Albanian (sq) as I was unable to find the source word list.

The naming and format of the personal dictionary is also different. However, ispell personal dictionaries can be imported using the `aspell-import` script. See Section 5.1

[Using aspell-import], page 21. The Ispell personal dictionary is simply a list of words while the Aspell one is a list of words with a header line. Thus it is also fairly easy to convert between the two. See [Format of the Personal Dictionary], page 24.

3.2.2.6 Missing Functionally

The only major area where Ispell is superior to Aspell is in the handling of multi character letters such as old ASCII encoding of accented characters.

However, Aspell can handle UTF-8 documents far better than Ispell can.

3.3 Using Aspell with other Applications

3.3.1 With Emacs and XEmacs

The easiest way to use Aspell with Emacs or XEmacs is to add this line:

```
(setq-default ispell-program-name "aspell")
```

to the end of your .emacs file.

For some reason version 3.0 of ispell.el (the lisp program that (x)emacs uses) want to reverse the suggestion list. To fix this add this line:

```
(setq-default ispell-extra-args '("--reverse"))
```

after the previous line in your .emacs file and it should solve the problem.

Ispell.el, version 3.1 (December 1, 1998) and better, has the list reversing problem fixed. You can find it at `http://www.kdstevens.com/~stevens/ispell-page.html`.

3.3.2 With LyX

Version 1.0 of LyX provides support for Aspell's learning from user's mistakes feature.

To use Aspell with LyX 1.0 either change the '`spell_command`' option in the '`.lyxrc`' file or use the `run-with-aspell` utility.

3.3.3 With VIM

To use Aspell in VIM you simply need to add the following line to your '`.vimrc`' file:

```
map ^T :w!<CR>:!aspell check %<CR>:e! %<CR>
```

I use *`Ctrl-T`* since that's the way you spell check in `pico`. In order to add a control character to your `.vimrc` you must type *`Ctrl-v`* first. In this case *`Ctrl-v Ctrl-t`*.

A more useful way to use Aspell, IMHO, is in combination with Newsbody (`http://www.image.dk/~byrial/newsbody/`) which is how I use it since VIM is my editor for my mailer and my news reader.

```
map ^T\\1\\2<CR>:e! %<CR>
map \\1 :w!<CR>
map \\2 :!newsbody -qs -n % -p aspell check \\%f<CR>
```

3.3.4 With Pine

To use Aspell in Pine simply change the option '`speller`' to

```
aspell --mode=email check
```

To change the '`speller`' option go to the main menu. Type `S` for *setup*, `C` for *config*, then `W` for *where is*. Type in `speller` as the word to find. The speller option should be highlighted now. Hit enter, type in the above line, and hit enter again. Then type `E` for *exit setup* and `Y` to save the change.

If you have a strong desire to check other people's comments change '`speller`' to

```
aspell check
```

instead which will avoid switching Aspell into email mode.

4 Customizing Aspell

The behavior of Aspell can be changed by any number of options which can be specified at either the command line, the environment variable `ASPELL_CONF`, a personal configuration file, or a global configuration file. Options specified on the command line override options specified by the environment variable. Options specified by the environment variable override options specified by either of the configuration files. Finally options specified by the personal configuration file override options specified in the global configuration file. Options specified in the environment variable `ASPELL_CONF`, a personal configuration file, or a global configuration file will take effect no matter how Aspell is used which includes being used by other applications.

Aspell has three basic type of options: *boolean*, *value*, and *list*.

Boolean options are either enabled or disabled, *value* options take a specific value, and *list* options can either have entries added or removed from the list.

4.1 Specifying Options

4.1.1 At the Command Line

All options specified at the command line have the following basic format:

`--`*`option`*`[=`*`value`*`]`

where the '`=`' can be replaced by whitespace.

However some options also have single letter abbreviations of the form:

`letter` `[`*`optional_whitespace value`*`]`

Any non-ASCII characters are expected to be in the encoding specified by the current locale.

To reset an option to the default value prefix the option with a '`reset-`' and don't specify a value.

4.1.1.1 Value options

To specify a value option simply specify the option with its corresponding value. For example to set the filter mode to TeX use '`--mode=tex`'.

If a value option has a single letter shortcut simply specify the single letter shortcut with its corresponding value. For example to use a the accented version of the American English dictionary use '`-d en_US-w_accents`'.

4.1.1.2 Boolean options

To enable a boolean option simply specify the option without any corresponding value, or prefix it with an '`enable-`'. For example to create a backup file use '`--backup`'. To disable a boolean option prefix the option name with a '`dont-`' or '`disable-`'. For avoid creating a backup file use '`--dont-backup`'. Boolean options can also be set directly like a value option where the value is either `"true"` or `"false"`, for example '`--backup=true`'.

If a boolean option has a single letter abbreviation simply give the letter corresponding to either enabling or disabling the option without any corresponding value. For example to consider run-together words legal use '`-C`' or to consider them illegal use '`-B`'

4.1.1.3 List options

To add a value to the list, prefix the option name with a '`add-`' and then specify the value to add. For example to add the URL filter use '`--add-filter url`'. To remove a value from a list option prefix the option name with a '`rem-`' and then specify the value to remove. For example to remove the URL filter use '`--rem-filter url`'. To remove all items from a list prefix the option name with a '`clear-`' without specify any value. For example to remove all filters use '`--clear-filter`'.

A list option can also be set directly, in which case it will be set to a single value. To directly set a list option to multiple values prefix the option name with a '`lset-`' and separate each value with a '`:`'. For example to use the URL and TeX filter use '`--lset-filter url:tex`'.

4.1.2 Via a Configuration File

Aspell can also accept options via a personal or global configuration file. The exact files to used are specified by the options '`per-conf`' and '`conf`' respectively but the personal configuration file is normally '`.aspell.conf`' located in the `HOME` directory and the global one is normally '`aspell.conf`' which is located in the '`etc`' directory which is normally '`/usr/etc`' or '`/usr/local/etc`'. To find out the particular values for your particular system use `aspell dump config`.

Each line of the configuration file has the format:

option [*value*]

There may be any number of spaces between the option and the value however it can only be spaces, i.e. there is no '`=`' between the option name and the value.

Comments may also be included by preceding them with a '`#`' as anything from a '`#`' to a newline is ignored. Blank lines are also allowed.

To include a literal '`#`' use '`\#`'. To include a literal '`\`' use '`\\`'. Any other non-alpha character can also be protected by a '`\`' if necessary.

Any non-ASCII characters are expected to be in UTF-8.

To reset an option to the default value prefix the option with a '`reset-`' and don't specify a value.

Values set in the personal configuration file override those in the global file. Options specified at either the command line or via an environment variable override those specified by either configuration file.

> **Note:** Filters and corresponding options also may be assembled inside a special meta filter file named '*metafilter*`.flt`'. A filter has to be loaded via adding a `add-filter` *filtername* line to the meta filter file before its options may be specified.

4.1.2.1 Value options

To specify a value option simply include the option followed by the corresponding option. For example to set the default language to German use '`lang german`'.

4.1.2.2 Boolean options

To specify a boolean option simply include the option followed by a '`true`' to enable it or a '`false`' to disable it. For example to allow run-together words use '`run-together true`'.

4.1.2.3 List options

To add a value to the list, prefix the option name with a '`add-`' and then specify the value to add. For example to add the URL filter use '`add-filter url`'. To remove a value from a list option prefix the option name with a '`rem-`' and then specify the value to remove. For example to remove the URL filter use '`rem-filter url`'. To remove all items from a list prefix the option name with a '`clear-`' without specifying any value. For example to remove all filters use '`clear-filter`'.

A list option can also be set directly, in which case it will be set to a single value. To directly set a list option to multiple values prefix the option name with a '`lset-`' and separate each value with a '`:`'. For example to use the URL and TeX filter use '`lset-filter url:tex`'. To include a literal '`:`' use '`\:`'.

4.1.3 Setting Options via an Environment Variable

The environment variable `ASPELL_CONF` may also be used and it overrides any options set in the configuration file. The format of the string is exactly the same as the configuration file except that semicolons ('`;`') are used instead of newlines.

4.2 The Options

The following is a list of available options broken down by category. Each entry has the following format:

option[**,***single-letter-abbreviation*]
(type) description

Where single letter options are specified as they would appear at the command line, ie with the preceding dash. Boolean single letter options are specified in the following format:

-<abbreviation to enable>|-<abbreviation to disable>

option is one of the following: *boolean*, *string*, *file*, *dir*, *integer*, or *list*.

String, *file*, *dir*, and *integer* types are all value options which can only take a specific type of value.

4.2.1 Dictionary Options

The following options may be used to control which dictionaries to use and how they behave (for more information see Section 5.2 [How Aspell Selects an Appropriate Dictionary], page 21):

master,-d *(string)* Base name of the dictionary to use. If this option is specified than Aspell will either use this dictionary or die.

dict-dir *(dir)* Location of the main word list.

lang *(string)* Language to use, it follows the same format of the `LANG` environment variable on most systems. It consists of the two letter ISO 639 language code and an optional two letter ISO 3166 country code after a dash or underscore. The default value is based on the value of the `LC_MESSAGES` locale.

size *(string)* The preferred size of the word list.

variety *(list)* Any extra information to distinguish two different words lists that have the same lang and size.

word-list-path
(list) Search path for word list information files.

personal,-p
(file) Personal word list file name.

repl *(file)* Replacements list file name.

extra-dicts
(list) Extra dictionaries to use.

dict-alias *(list)* create dictionary aliases. Each entry has the form '*`from to`*'. Will override any system dictionaries that are present.

4.2.2 Encoding Options

These options control the encoding the document is expected to be in and how it is displayed.

encoding *(string)* The encoding the input text is in. Valid values include, but not limited to, '`iso-8859-*`', '`utf-8`', '`ucs-2`', '`ucs-4`'. When using the Aspell utility the default encoding is based on the current locale. Thus if your locale currently uses the '`utf-8`' encoding than everything will be in UTF-8. The '`ucs-2`' and '`ucs-4`' encodings are intended to be used by other programs using the Aspell library and is not supported by the Aspell utility.

normalize *(boolean)* Perform Unicode normalization. Enabled by default.

norm-strict
(boolean) Avoid lossy conversions when normalizing. Lossy conversions includes compatibility mappings such as splitting the letter '`OE`' (U+152) into '`O`' and '`E`' (when the combined letter is not available), and mappings which will remove accents. Disabled by default except when creating dictionaries.

norm-form
(string) The normalization form the output should be in. This option primarily effects the normalization form of the suggestions as when spell checkering as the actual text is unchanged unless there is an error. Valid values are '`none`', '`nfd`' for fully decomposition (Normalization Form D), '`nfc`' for Normalization Form C, or '`comp`' for fully composed. '`comp`' is like '`nfc`' except that *full* composition is used rather than *canonical* composition. The '`normalize`' option must be enabled for this option to be used.

norm-required
(boolean) Set to true when the current language requires Unicode normalization. This is generally the case when private use characters are used internally by Aspell or when Normalization Form C is not the same as full composition.

4.2.3 Checker Options

These options control the behavior of Aspell when checking documents.

ignore,-W *(integer)* Ignore words with N characters or less

ignore-repl
(boolean) Ignore commands to store replacement pairs.

save-repl *(boolean)* Save the replacement word list on save all.

keyboard *(file)* The base name of the keyboard definition file to use (see Section 4.4.3 [Notes on Typo-Analysis], page 19)

sug-mode *(mode)* Suggestion mode = '`ultra`' | '`fast`' | '`normal`' | '`slow`' | '`bad-spellers`' (see Section 4.4.4 [Notes on the Different Suggestion Modes], page 20)

ignore-case
(boolean) Ignore case when checking words.

ignore-accents
(boolean) Ignore accents when checking words – *currently ignored.*

4.2.4 Filter Options

These options modify the behavior of the Aspell filter interface in general (for more information see see Section 4.4.1 [Notes on Various Filters and Filter Modes], page 16).

filter *(list)* filters to use

filter-path *(list)* Where to look when loading filter and filter modes.

mode *(string)* Sets the filter mode. Possible values include, but not limited to, '`none`', '`url`', '`email`', '`sgml`', or '`tex`'. (The shortcut options '`-e`' may be used for email, '`-H`' for HTML, or '`-t`' for TEX).

These options belong to filters packaged along with Aspell standard distribution. These options may be prefixed by the keyword `f-` in order to explicitly indicate that they are options recognized by a filter and not by Aspell itself.

4.2.4.1 email

This filter hides quoting characters and email preamble and other parts of an email which need not to be spell checked.

email-quote
(list) Email quote characters.

email-margin
(integer) The number of characters that can appear before the quote character

4.2.4.2 html

This filter converts an HTML source file into a format which eases spell checking of HTML texts by Aspell.

html-check
(list) HTML attributes to always check, such as alt= (alternate text).

html-skip *(list)* HTML tags to always skip the contents of, such as <script>.

4.2.4.3 sgml

This filter is identical to the HTML filter except that it options has different default values which are currently the empty list.

4.2.4.4 tex/latex

This filter hides all LaTeX commands and corresponding parameters not being readable text in LaTeX output from Aspell.

tex-command
: *(list)* TEX commands

tex-check-comments
: *(boolean)* check TEX comments

4.2.4.5 texinfo

This filter hides all Texinfo command from Aspell. It can also hide Texinfo parameters and environments not corresponding to readable text.

texinfo-ignore
: *(list)* Texinfo command to ignore the parameters of.

texinfo-ignore-env
: *(list)* Texinfo environments to ignore.

4.2.4.6 context

This filter can be used to spell check source codes, HTML sources and other texts which consist of different contexts. These contexts must be separated by pairs of unique delimiters. The different contexts may not be dependent upon each other except for initial context which is assumed if not any other context applies.

context-visible-first
: *(boolean)* Switches the context which should be visible to Aspell. Per default the initial context is assumed to be invisible as one would expect when spell checking source files of programs where relevant parts are contained in string constants and comments but not in the remaining code. If set to true the initial context is visible while the delimited ones are hidden.

add|rem-context-delimiters
: *(list)* Add or remove pairs of delimiters. This allows you to specify the character, or sequences of characters, which should be used to switch contexts and therefore have to be escaped by '\' if they should appear literally. The two delimiting chars belonging to one pair have to be separated by a space character. If multiple pairs are specified by one '`add|rem-context-delimiters`' call the different pairs have to be separated by a literal comma. Per default the delimiters are set to C/C++ comment and string constant delimiters. If the end of line delimits a context than this has to be indicated by the literal '`\0`' string.

4.2.5 Run-together Word Options

These may be used to control the behavior of run-together words (for more information see Section 7.7 [Controlling the Behavior of Run-together Words], page 40):

run-together,-C|-B
(boolean) consider run-together words legal

run-together-limit
(integer) maximum numbers that can be strung together

run-together-min
(integer) minimal length of interior words

4.2.6 Miscellaneous Options

Miscellaneous other options that don't fall under any other category

conf *(file)* main configuration file

conf-dir *(dir)* location of main configuration file

data-dir *(dir)* location of language data files

local-data-dir
(dir) alternative location of language data files. This directory is searched before '`data-dir`'. It defaults to the same directory the actual main word list is in (which is not necessarily '`dict-dir`').

home-dir *(dir)* location for personal files

per-conf *(file)* personal configuration file

prefix *(dir)* prefix directory

set-prefix *(boolean)* set the prefix based on executable location (only works on WIN32 and when compiled with '`--enable-win32-relocatable`')

4.2.7 Aspell Utility Options

backup,-b|-x
(boolean) Create a backup file by appending '`.bak`' to the file name. (Only applies when the command is `check`).

time *(boolean)* Time load time and suggest time in `pipe` mode.

byte-offsets
(boolean) Use byte offsets instead of character offsets in `pipe` mode.

reverse *(boolean)* Reverse the order of the suggestions list in `pipe` mode.

keymapping
(string) the keymapping to use. Either '`aspell`' for the default mapping or '`ispell`' to use the same mapping that the Ispell utility uses.

guess *(boolean)* make possible root/affix combinations not in the dictionary in `pipe` mode.

suggest *(boolean)* Suggest possible replacements in `pipe` mode. If false Aspell will simply report the misspelling and make no attempt at suggestions or possible corrections.

4.3 Dumping Configuration Values

To find out the current value of all the options use the command `aspell dump config`. This will dump the current configuration to standard output. The format of the contents dumped is such that it can be used as either the global or personal configuration file.

To find out the current value of a particular option use `aspell config` *`option`*. This will print out the value of *option* to `stdout` and nothing else.

4.4 Notes on Various Options

4.4.1 Notes on Various Filters and Filter Modes

Aspell now has filter support. You can either select from individual filters or choose a filter mode. To select a filter mode use the '`mode`' option. You may choose from '`none`', '`url`', '`email`', '`sgml`', '`ccpp`', '`tex`' and any other available on your system. The default mode is '`url`'. Individual filters can be added with the option '`add-filter`' and removed with the '`rem-filter`' option. The currently available filters are '`url`', '`email`', '`sgml`' and '`tex`', '`latex`' (alias for '`tex`'), '`nroff`', '`context`', as well as a bunch of filters which translate the text from one format to another.

To check which filters are available use `aspell dump filters`. To check which filter modes are available use `aspell dump modes`. The `aspell help` command will also list all available filter and filter modes.

4.4.1.1 None Filter Mode

The '`none`' mode is exactly what it says. It turns off all filters.

4.4.1.2 URL Filter

The '`url`' filter/mode skips over URLs, host names, and email addresses. Because this filter is almost always useful and rarely does any harm it is enabled in all modes except '`none`'. To turn it off either select the '`none`' mode or use '`rem-filter`' option *after* the desired mode is selected.

4.4.1.3 Email Filter

The '`email`' filter mode skips over quoted text. It currently does not support skipping over headers however a future version should. In the meantime I suggest you use Aspell with Newsbody which can be found at `http://home.worldonline.dk/~byrial/newsbody/`. The option '`email-skip`' controls the number of characters that can appear before the email quote character, the default is 10. The option '`add|rem-email-quote`' controls the characters that are considered quote characters, the defaults are '`>`' and '`|`'.

4.4.1.4 SGML Filter

The SGML filter allows you to spell check SGML, HTML, XHTML, and XML files. In most cases everything within a tag '`<tag attrib=value attrib2="a whole sentence">`' will be skipped by the spell checker. The SGML/HTML/XML that Aspell supports is a slight superset of most DTDs (Document Type Definitions) and can spell check the often non-conforming HTML found on the web.

Two configuration options, '`sgml-skip`' and '`sgml-check`', allow you to control what is spell checked. The tag and attribute names specified are case insensitive.

sgml-skip This is a list of tags whose contents will also be skipped by the spell checker. For example, if you wish to leave a misspelling in a document and not have them flagged as misspellings, you could surround them with a <nospellcheck> tag:

```
<TD><FONT size=2><NOSPELLCHECK>leviosa</NOSPELLCHECK>
is what Mr. Potter said</FONT></TD>
```

And put that word in the skip config directive:

```
add-sgml-skip nospellcheck
```

sgml-check

This is a list of attributes whose values you do want spell checked. By default, 'alt' (<img> alternate text) is a member of the check list since it is text that is seen by a web page viewer. You may also want 'value' to be on the check list since that is the text put on buttons:

```
add-sgml-check value
```

In this case '`<input type=button value="Donr">`' will be flagged as a misspelling.

This filter will also translate SGML characters of the form '`&#num;`'. Other SGML characters such as '`&`' will simply be skipped over so that the word '`amp`', for example, will not be spell checked. Eventually full support for properly translating SGML characters will be added.

4.4.1.5 HTML Filter

The '`html`' filter is like the SGML Filter Mode but specialized for HTML. By default, 'script' and 'style' are members of the skip list in HTML mode.

4.4.1.6 TEX/LaTeX Filter

The '`tex`' (all lowercase) filter mode skips over TEX commands and parameters and/or options to certain commands. It also skips over TEX comments by default. The option '`[dont-]tex-check-comments`' controls whether or not Aspell will skip over TEX comments. The option '`add|rem-tex-command`' controls which TEX commands should have certain parameters and/or options also skipped over. Commands that are not specified will have all their parameters and/or options checked. The format for each item is

```
<command> <a list of p,P,o and Os>
```

The first item is simply the command name. The second item controls which parameters to skip over. A 'p' skips over a parameter while a 'P' doesn't. Similar an 'o' will skip over an optional parameter while a 'O' doesn't. The first letter on the list will apply to the first parameter, the second letter will apply to the second parameter etc. If there are more parameters than letters Aspell will simply check them as normal. For example the option

```
add-tex-command rule pp
```

will skip over the first two parameters of the `rule` command while the option

```
add-tex-command foo Pop
```

will *check* the first parameter of the `foo` command, skip over the next optional parameter, if it is present, and will skip over the second parameter — even if the optional parameter is not present — and will check any additional parameters.

A '`*`' at the end of the command is simply ignored. For example the option

```
enlargethispage p
```

will ignore the first parameter in both '`enlargethispage`' and '`enlargethispage*`'.

To remove a command simple use the '`rem-tex-command`' option. For example

```
rem-tex-command foo
```

will remove the command foo, if present, from the list of TEX commands.

The TeX filter mode is also available via '`latex`' alias name.

4.4.1.7 Texinfo Filter

The '`texinfo`' filter allows you to spell check Texinfo files. It will skip over any Texinfo commands and their parameters when appropriate. It will also skip over some Texinfo environments such as `example`. The list option '`texinfo-ignore`' controls which commands to ignore the parameters of and the list option '`texinfo-ignore-env`' controls which Texinfo environments to ignore.

The Texinfo filter has special code to deal with the `@table` and related commands. It will apply the formatting command to each of the `@item` or `@itemx` commands just like Texinfo will. This means that if the formatting command is `@code` and and the `@code` command is a member of the '`texinfo-ignore`' option than the Texinfo filter will ignore the parameter of the `@item` command as if the parameter was also the parameter of the `@code` command.

The Texinfo filter will also skip over the '`\input texinfo`' line.

4.4.1.8 Nroff Filter

The '`nroff`' filter mode allows you to check the spelling of Nroff documents. The mode is enabled by giving '`--add-filter=nroff`' or '`-n`' command line option to `aspell`. It is also automatically enabled if the first three characters of the file being checked are `.\"` (a `nroff` comment marker) or the file name ends in a one of the following suffixes:

- single decimal digit from '`0`' to '`9`'
- letter '`n`'
- '`tmac`'

This filter mode skips following `nroff` language elements:

- Comments
- Requests
- Names of `nroff` registers (both traditional two-letter names and GNU nroff long names)
- Arguments to the following requests: `ds`, `de`, `nr`, `do`, `so`.
- Arguments to font switch (`\f`) and size switch (`\s`) escapes
- Arguments to extended charset escape in both traditional (`\(`) and extended (`\[comp1 comp2 ...]`) form.

4.4.1.9 Context Filter

The *context* filter allows Aspell to distinguish between visible and invisible contexts. The visible ones will be spell checked and the invisible ones will be ignored. The contexts are distinguished by the fact that the visible/invisible ones are delimited by specific and unique delimiter characters or character sequences. Whether the delimited contexts should be visible or invisible only stated by the value of the '`[dont-]context-visible-first`' option and not by the delimiters.

The context delimiters are specified as pairs of delimiters via the '`add|rem-context-delimiters`' option. The delimiters enclosing a specific context are specified as a space separated pair. If more than one delimiter pair is specified by one call of '`add|rem-context-delimiters`' they have to be combined to a comma separated list. To indicate that a context is always closed by end of line use `\0` sequence as closing delimiter.

4.4.1.10 Ccpp Filter Mode

The '`ccpp`' filter mode will limit spell checking to C/C++ comments and string literals. Any code in between will be left alone.

4.4.2 Notes on the Prefix Option

The '`prefix`' option is there to allow Aspell to easily be relocated. Changing '`prefix`' will change all directory names relative to the new prefix that are not explicitly set. For example if '`prefix`' was '`/usr/local/aspell`' and '`dict-dir`' has a default value of '`/usr/local/aspell/dict`' than changing '`prefix`' to '`/opt/aspell`' will also change the default value of '`dict-dir`' to '`/opt/aspell/dict`'. Note that modifying '`prefix`' will only effect the default compiled in values of directories. If a directory option is explicitly given a value than changing the value of '`prefix`' has no effect on that directory option.

4.4.3 Notes on Typo-Analysis and the Keyboard Definition File

Aspell .33 and better will, in general, give a higher priority to certain misspellings which are likely to be due to typos such as `teh` instead of `the` or `hapoy` instead of `happy`. However in order to do this well Aspell needs to know the layout of the keyboard via the keyboard definition file. The keyboard definition file simply identifies the keys on the keyboard and which of them are right next to each other. It has an extension of '`.kbd`' and all non-ASCII characters are expected to be in UTF-8.

To identify a key use:

```
key base other ...
```

where *base* is the base character that the key types, and *other* are other keys that the key can produce. For example

```
key a A á Á
```

It generally is only necessary to list keys which type more than one distinct letter as Aspell can derive the rest from the language data file. For example, it is not necessary to include the previously mentioned key.

To identify two keys as being right next to each other simply list the type keys right after each other. For example the line:

```
as
```

will indicate that '`a`' and '`s`' are right next to each other. If '`as`' is listed as a entry it is not necessary to list '`sa`' as an entry as that will be done automatically. Also by *right next to each other* I mean two keys that are close enough together that it is easy to type one instead of the other. On most keyboards this means keys that are to the left or to the right of each other and *not* keys that are below or above it.

The default for this option is normally '`standard`'. However the default can be changed via the language data file. The normal default, '`standard`', should work well for most QWERTY like keyboard layouts. It may need minor adjusting for foreign keyboards. The '`dvorak`' option can be used for a Dvorak layout.

4.4.4 Notes on the Different Suggestion Modes

In order to understand what these suggestion modes do, a basic understanding of how Aspell works is required. For that see Section A.1 [Aspell Suggestion Strategy], page 42.

The suggestion modes are as follows.

ultra This method will use the fastest method available to come up with decent suggestions. This currently means that it will look for soundslikes within one edit distance. This method will also use the replacement table if one is available. In this mode Aspell gets about 87% of the words from my small test kernel of misspelled words. (Go to `http://aspell.net/test` for more info on the test kernel as well as comparisons of this version of Aspell with previous versions and other spell checkers.)

fast This method is currently identical to '`ultra`'.

normal This mode will use what ever method is necessary to return good suggestions in most cases in a reasonable amount of time. This currently means it will looks for soundslikes within two edit distance apart. This mode gets 93% of the words.

slow Like '`normal`' except that "reasonable amount of time" is not a consideration. In most cases it will return the same results as '`normal`'. The biggest difference is that it will try an ngram scan if the normal methods of finding a suggestion fail.

bad-spellers
This method is like '`slow`' but is tailored more for the bad speller, where as the other modes are tailored more to strike a good balance between typos and true misspellings. This mode never performs typo-analysis and returns a *huge* number of words for the really bad spellers who can't seem to get the spelling anything close to what it should be. If the misspelled word looks anything like the correct spelling it is bound to be found *somewhere* on the list of 100 or more suggestions. This mode gets 98% of the words.

If jump tables were not used then the '`normal`' option is identical to '`fast`'. And the '`slow`' option is identical to the '`normal`' if jump tables were used.

5 Working With Dictionaries

5.1 Using `aspell-import`

The `aspell-import` Perl script will look for old personal dictionaries and will import them into GNU Aspell. It will look for both Ispell and Aspell ones. To use it, just run it from the command prompt. If you get an error about '`/usr/bin/perl`' not being found, then instead try `perl` *`bindir`*`/aspell-import`. When running the script if you get a message like:

```
Error: No word lists can be found for the language "de".
```

This means that you have not installed support for the given language, in this case `de` for German. To rectify the situation download and install a dictionary designed to work with GNU Aspell 0.50 or better.

5.2 How Aspell Selects an Appropriate Dictionary

If the '`master`' option is set in any fashion (via the command line, the `ASPELL_CONF` environment variable, or a configuration file) Aspell will look for a dictionary of that name. If one could not be found, it will complain.

Otherwise it will use the value of the '`lang`' option to search for an appropriate dictionary. If more than one dictionary is found for the given language string then it will look for a dictionary with a matching variety if the '`variety`' option is set. If it is not set it will look for a dictionary without a variety. If after matching the '`lang`' and '`variety`' there is still more than one dictionary available it will find one with the size closest to the value of the '`size`' option. The default size is 60. If Aspell cannot find a dictionary based on the '`lang`' option then it will give up and complain.

If the '`lang`' option is not explicitly set its value will be based on the `LC_MESSAGES` locale. This locale is generally taken from the `LC_MESSAGES` environment variable or the `LANG` environment variable if `LC_MESSAGES` is not set. However, if Aspell is being used as a library from within another program which already explicitly set the locale then it will use the locale of the library rather than the environment variables. If Aspell cannot determine the language from the `LC_MESSAGES` locale than it will default to `en_US`.

The list option '`dict-alias`' can be used to influence which dictionary is selected by creating an alias from one dictionary name to another. This option is most useful when there is more than one dictionary for a given language. For example '`add-dict-alias en_US en_US-w_accents`' will cause Aspell to choose the accented version of the American English dictionary instead of the non-accented version. To add an alias use:

```
add-dict-alias NAME VAL
```

5.3 Listing Available Dictionaries

For a list of available dictionaries use the command `aspell dump dicts`. This will form a list of dictionaries that Aspell will search when a dictionary is not specifically given.

5.4 Dumping the Contents of the Word List

The dump command in `aspell` will simply dump the contents of a word list to '`stdout`' in a format that can be read back in with `aspell create`.

If no word list is specified the command will act on the default one. For example the command

```
aspell dump personal
```

will simply dump the contents of the current personal word list to '`stdout`'.

5.5 Creating an Individual Word List

To create an individual main word list from a list of words use the command

```
aspell --lang=lang create master ./base < wordlist
```

where *base* is the name of the word list and *wordlist* is the list of words separated by white space. The name of the word list will automatically be converted to all lowercase. The `./` is important because without it Aspell will create the word list in the normal word list directory. If you are trying to create a word list in a language other than English check the Aspell '`data-dir`' (usually '`/usr/share/aspell`', use `aspell dump config` to find out what it is on your system) to see if a language data file exists for your language. If not you will need to create one. For more information on using Aspell with other languages Chapter 7 [Adding Support For Other Languages], page 31.

This will create the file '*base*' in the current directory. To use the new word list copy the file to the normal word list directory (use `aspell config` to find out what it is) and use the option '`--master=base`'.

During the creating of the dictionary you may get a number of warnings or errors about invalid words or affixes. By default Aspell will skip any invalid words and remove invalid affixes. If you rather that Aspell simply accepts all words given then the option '`--dont-validate-words`' can be specified. To avoid checking if affixes are valid use the option '`--dont-validate-affixes`'. However, rather than disable checking, it is preferable to clean the input word list. This can be done by using the command

```
aspell --local-data-dir=./ --lang=lang clean < wordlist > result
```

which will clean the word list and output the results to *result*. By default it will remove invalid characters from the beginning and end of a word before resorting to skipping the word. If you rather it just skip the words than add the keyword strict:

```
aspell --local-data-dir=./ --lang=lang clean strict < wordlist > result
```

The option '`--clean-words`' can be be added when creating a dictionary if you want Aspell to remove invalid characters from the beginning and end of a word like the `"clean"` command does. In addition the options '`--dont-skip-invalid-words`' and '`--dont-clean-affixes`' can be specified to turn the warnings into errors.

The compiled dictionary file are endian order dependent. When a dictionary is loaded the endian order is checked. Please do not distribute the compiled dictionaries unless you are only distributing them for a particular platform such as you would a binary.

Aspell is now also able to use special `multi` dictionaries. for more information Section 5.2 [How Aspell Selects an Appropriate Dictionary], page 21.

A personal and replacement word list can be created in a similar fashion.

5.5.1 Format of the Replacement Word List

The replacement word list has each replacement pair on its own line in the following format

```
misspelled_word correction
```

5.6 Working With Affix Info in Word Lists

5.6.1 The Munch Command

The `munch` command takes a list of words from standard input and outputs a list of possible root words and affixes. The root may, however, be invalid as it does not check them against the exiting dictionary. For example the command:

```
echo brother | aspell -l en munch
```

produces

```
brother broth/R brothe/R
```

5.6.2 The Expand Command

The `expand` command is the reverse of `munch`, it expands affix flags to produce a list of words. For example:

```
echo both/R | aspell -l en expand
```

produces

```
both bother
```

It formal usage is:

```
aspell expand [level] [limit]
```

Where *level* is the expansion level. Valid values are between 1 and 3. Level 1 is the default if not otherwise specified. Level 2 causes the original root/affix to be included, for example:

```
both/R both bother
```

Level 3 causes multiple lines to be printed, one for each generated word, with the original root/affix combination followed by the word it creates:

```
both/R both
both/R bother
```

Levels larger than 3 may also be supported, but should not used as they may eventually be removed.

If a *limit* parameter is given than only expansions which effect the first *limit* letters will be expanded. If a base word is not completely expanded for a given affix flag that flag will be left on the word. Note that prefixes are always expanded.

5.6.3 The Munch-list Command

The `munch-list` command will reduce the size of word list via affix compression. It will reduce a list of words to a minimal (or close to it) set of roots and affixes that will match the same list of words. The list of words is read from standard input and the result, the "munched" list, is written to standard out. It's usage is:

```
aspell munch-list [keep] [single|multi] [simple] < infile > outfile
```

where '`simple`', '`single`', '`multi`', and '`keep`' are literal values.

The default algorithm used should give near optimum results. In some cases the set of words returned is, provably, the minimum number possible. In the typical case the number of words returned is within 1% of the optimum number.

By default Aspell will remove redundant affix flags. The '`keep`' flag will avoid removing them, which can be useful if you want to include all possible expansions for each base word.

When cross products are involved it may be beneficial to list a base word more than once. Unfortunately, the current version of Aspell can not correctly handle multiple base words in a dictionary. Therefore, the current default behavior is to only include the one with the most expansions. All of them can be included via the '`multi`' flag. Once aspell is able to handle multiple base words the default will be to include them all. The '`single`' flag can be used to only include one of them.

The '`simple`' flag will select an alternate faster algorithm. This algorithm is very similar to the `munch` command distributed with MySpell (the Open Office spell checker), however, it doesn't give nearly as good results. It does okay for the English word list but not for some other languages such as German; the normal algorithm reduced a list of 312,002 German words to 79,420 base words while the simple algorithm only reduced it to 115,927 words. This algorithm may disappear in a future version of Aspell.

5.7 Format of the Personal and Replacement Dictionaries

5.7.1 Format of the Personal Dictionary

The personal dictionary generally has a filename of the form:

```
.aspell.lang.pws
```

And the file itself contains two parts. The first part is a header line of the form:

```
personal_ws-1.1 lang num [encoding]
```

where *num* is the number of words in the list. This number is only used as a hint, and thus does not have to be accurate. When creating a new dictionary it is perfectly acceptable for *num* to be 0. The *encoding* is optional and specifies the encoding of the word list. If it is left out the encoding is expected to be in the default encoding for the language as specified by the '`data-encoding`' option. See [data-encoding], page 31.

The second part of simply a word list with one word per line.

5.7.2 Format of the Personal Replacement Dictionary

The personal replacement dictionary generally has a filename of the form:

```
.aspell.lang.prepl
```

And the file itself contains two parts. The first part is a header line of the form:

```
personal_repl-1.1 lang num [encoding]
```

where *num* is currently unused and thus always 0. As with the personal dictionary the *encoding* is optional.

The second part simply a list of replacements with one replacement per line with each replacement pair has the following format:

```
misspelled_word correction
```

5.8 Using Multi Dictionaries

As with previous versions of Aspell you can specify the main dictionary to use via the '`-d`' or '`--master`' option. However as of Aspell .32 you can now also:

1. Specify more than word list to use with the '`extra-dicts`' option.
2. Specify special *multi* dictionaries.

The '`extra-dicts`' is a list option. To add a dictionary use '`add-extra-dicts`' or to remove a dictionary from the list use '`rem-extra-dicts`'.

A *multi* dictionary is a special file which is basically a list of dictionary files to use. A *multi* dictionary must end in '`.multi`' and has roughly the same format as a configuration file with the only accepted key being '`add`'.

5.9 Dictionary Naming

In order for Aspell to be able to correctly recognize a dictionary based on the setting of the `LANG` environment variable the dictionaries need to be located somewhere Aspell can find them and they need to be *multi* dictionaries. Where aspell looks for dictionaries depends on the value of the '`dict-dir`' and '`word-list-path`' option.

'`dict-dir`' is generally '*prefix*`/lib/aspell`', and '`word-list-path`' is generally empty.

Each dictionary that you expect Aspell to be able to find needs to have a name in the following format:

language`[_`*region*`][-`*variety*`][-`*size*`].multi`

where *language* is the two letter language code, *region* is the two letter region code, *variety* is any extra information to distinguish the word list from other ones with the same language and spelling. Multiple varieties can be used by separating them with a '-'. Finally, *size* is the size of the dictionary. If no size is specified that the default size of 60 will be assumed.

For example:

```
en.multi
en_US.multi
en-medical.multi
en-medical-85.multi
en-85.multi
de.multi
```

5.10 AWLI files

In order for Aspell to find dictionaries that are located in odd places or not named according to Section 5.9 [Dictionary Naming], page 25, an AWLI file needs to be created for the dictionary and located in some place where Aspell can find it.

Each AWLI file has a name in the following format:

language`[`*region*`][-`*variety*`][-`*size*`]-`*module*`.awli`

where the names have the same meaning as in Section 5.9 [Dictionary Naming], page 25, and *module* is the speller module to use, which should be set to *default* for now since there is only one speller module.

Each '`awli`' file for an Aspell word list should then contain exactly one line which contains the full path of the main word list.

6 Writing programs to use Aspell

There are two main ways to use Aspell from within your application. Through the external C API or through a pipe. The internal Aspell API can be used directly but that is not recommended as the actual Aspell API is constantly changing.

6.1 Through the C API

The Aspell library contains two main classes and several helper classes. The two main classes are `AspellConfig` and `AspellSpeller`. The `AspellConfig` class is used to set initial defaults and to change spell checker specific options. The `AspellSpeller` class does most of the real work. The `C API` is responsible for managing the dictionaries, checking if a word is in the dictionary, and coming up with suggestions among other things. There are many helper classes the important ones are `AspellWordList`, `AspellMutableWordList`, `Aspell*Enumeration`. The `AspellWordList` classes is used for accessing the suggestion list, as well as the personal and suggestion word list currently in use. The `AspellMutableWordList` is used to manage the personal, and perhaps other, word lists. The `Aspell*Enumeration` classes are used for iterating through a list.

6.1.1 Usage

To use Aspell your application should include '`aspell.h`'. In order to ensure that all the necessary libraries are linked in libtool should be used to perform the linking. When using libtool simply linking with `-laspell` should be all that is necessary. When using shared libraries you might be able to simply link `-laspell`, but this is not recommended. This version of Aspell uses the CVS version of libtool however released versions of libtool should also work.

When your application first starts you should get a new configuration class with the command:

```
AspellConfig * spell_config = new_aspell_config();
```

which will create a new `AspellConfig` class. It is allocated with `new` and it is your responsibility to delete it with `delete_aspell_config`. Once you have the config class you should set some variables. The most important one is the language variable. To do so use the command:

```
aspell_config_replace(spell_config, "lang", "en_US");
```

which will set the default language to use to American English. The language is expected to be the standard two letter ISO 639 language code, with an optional two letter ISO 3166 country code after an underscore. You can set the preferred size via the '`size`' option, any extra info via the '`variety`' option, and the encoding via the '`encoding`' option. Other things you might want to set is the preferred spell checker to use, the search path for dictionaries, and the like — see Section 4.2 [The Options], page 11, for a list of all available options.

Whenever a new document is created a new `AspellSpeller` class should also be created. There should be one speller class per document. To create a new speller class use the `new_aspell_speller` and then cast it up using `to_aspell_speller` like so:

```
AspellCanHaveError * possible_err = new_aspell_speller(spell_config);
AspellSpeller * spell_checker = 0;
```

```
if (aspell_error_number(possible_err) != 0)
  puts(aspell_error_message(possible_err));
else
  spell_checker = to_aspell_speller(possible_err);
```

which will create a new `AspellSpeller` class using the defaults found in `spell_config`. To find out which dictionary is selected the '`lang`', '`size`', and '`variety`' options may be examined. To find out the exact name of the dictionary the '`master`' option may be examined as well as the '`master-flags`' options to see if there were any special flags that were passed on to the module. The '`module`' option way also be examined to figure out which speller module was selected, but since there is only one this option will always be the same.

If for some reason you want to use different defaults simply clone `spell_config` and change the setting like so:

```
AspellConfig * spell_config2 = aspell_config_clone(spell_config);
aspell_config_replace(spell_config2, "lang","nl");
possible_err = new_aspell_speller(spell_config2);
delete_aspell_config(spell_config2);
```

Once the speller class is created you can use the `check` method to see if a word in the document is correct like so:

```
int correct = aspell_speller_check(spell_checker, word, size);
```

word is expected to be a `const char *` character string. If the encoding is set to be `ucs-2` or `ucs-4` *word* is expected to be a cast from either `const u16int *` or `const u32int *` respectively. `u16int` and `u32int` are generally `unsigned short` and `unsigned int` respectively. *size* is the length of the string or `-1` if the string is null terminated. If the string is a cast from `const u16int *` or `const u32int *` then *size* is the amount of space in bytes the string takes up after being cast to `const char *` and not the true size of the string. `sspell_speller_check` will return `0` if it is not found and non-zero otherwise.

If the word is not correct, then the `suggest` method can be used to come up with likely replacements.

```
AspellWordList * suggestions = aspell_speller_suggest(spell_checker,
                                                      word, size);
AspellStringEnumeration * elements = aspell_word_list_elements(suggestions);
const char * word;
while ( (word = aspell_string_enumeration_next(aspell_elements)) != NULL )
{
  // add to suggestion list
}
delete_aspell_string_enumeration(elements);
```

Notice how `elements` is deleted but `suggestions` is not. The value returned by `suggestions` is only valid to the next call to `suggest`. Once a replacement is made the `store_repl` method should be used to communicate the replacement pair back to the spell checker (for the reason, see Section 6.3 [Notes on Storing Replacement Pairs], page 30). Its usage is as follows:

```
aspell_speller_store_repl(spell_checker, misspelled_word, size,
                          correctly_spelled_word, size);
```

If the user decided to add the word to the session or personal dictionary the the word can be be added using the `add_to_session` or `add_to_personal` methods respectively like so:

```
aspell_speller_add_to_session|personal(spell_checker, word, size);
```

It is better to let the spell checker manage these words rather than doing it yourself so that the words have a chance of appearing in the suggestion list.

Finally, when the document is closed the `AspellSpeller` class should be deleted like so:

```
delete_aspell_speller(spell_checker);
```

6.1.2 API Reference

Methods that return a boolean result generally return `false` on error and `true` otherwise. To find out what went wrong use the `error_number` and `error_message` methods. Unless otherwise stated methods that return a `const char *` will return `NULL` on error. In general, the character string returned is only valid until the next method which returns a `const char *` is called.

For the details of the various classes please see the header files. In the future I will generate class references using some automated tool.

6.1.3 Examples

Two simple examples are included in the examples directory. The `example-c` program demonstrates most of the Aspell library functionality and the `list-dicts` lists the available dictionaries.

6.1.4 Notes About Thread Safety

Aspell should be thread safe, when used properly, as long as the underlying compiler, C and C++ library is thread safe. Aspell objects, including the AspellSpeller class, should not be used by multiple threads unless they are protected by locks or it is only accessed by read-only methods. A method is read-only only if a `const` object is passed in. Many methods that seam to be read-only are not because they may store state information in the object.

6.2 Through A Pipe

When given the `pipe` or `-a` command, Aspell goes into a pipe mode that is compatible with `ispell -a`. Aspell also defines its own set of extensions to Ispell pipe mode.

6.2.1 Format of the Data Stream

In this mode, Aspell prints a one-line version identification message, and then begins reading lines of input. For each input line, a single line is written to the standard output for each word checked for spelling on the line. If the word was found in the main dictionary, or your personal dictionary, then the line contains only a '`*`'.

If the word is not in the dictionary, but there are suggestions, then the line contains an '`&`', a space, the misspelled word, a space, the number of near misses, the number of characters between the beginning of the line and the beginning of the misspelled word, a colon, another space, and a list of the suggestions separated by commas and spaces.

Finally, if the word does not appear in the dictionary, and there are no suggestions, then the line contains a '`#`', a space, the misspelled word, a space, and the character offset from the beginning of the line. Each sentence of text input is terminated with an additional blank line, indicating that Aspell has completed processing the input line.

These output lines can be summarized as follows:

OK: `*`
Suggestions: `& original count offset: miss, miss, ...`
None: `# original offset`

When in the '`-a`' mode, Aspell will also accept lines of single words prefixed with any of '`*`', '`&`', '`@`', '`+`', '`-`', '`~`', '`#`', '`!`', '`%`', or '`^`'. A line starting with '`*`' tells Aspell to insert the word into the user's dictionary. A line starting with '`&`' tells Aspell to insert an all-lowercase version of the word into the user's dictionary. A line starting with '`@`' causes Aspell to accept this word in the future. A line starting with '`+`', followed immediately by a valid mode will cause Aspell to parse future input according the syntax of that formatter. A line consisting solely of a '`+`' will place Aspell in TEX/LaTeX mode (similar to the '`-t`' option) and '`-`' returns Aspell to its default mode (which is Nroff unless otherwise specified). (but these commands are obsolete). A line '`~`', is ignored for Ispell compatibility. A line prefixed with '`#`' will cause the personal dictionaries to be saved. A line prefixed with '`!`' will turn on terse mode (see below), and a line prefixed with '`%`' will return Aspell to normal (non-terse) mode. Any input following the prefix characters '`+`', '`-`', '`#`', '`!`', '`~`', or '`%`' is ignored, as is any input following. To allow spell-checking of lines beginning with these characters, a line starting with '`^`' has that character removed before it is passed to the spell-checking code. It is recommended that programmatic interfaces prefix every data line with an uparrow to protect themselves against future changes in Aspell.

To summarize these:

`*word`	Add a word to the personal dictionary
`&word`	Insert the all-lowercase version of the word in the personal dictionary
`@word`	Accept the word, but leave it out of the dictionary
`#`	Save the current personal dictionary
`~`	Ignored for Ispell compatibility.
`+`	Enter TEX mode.
`+mode`	Enter the mode specified by *mode*.
`-`	Enter the default mode.
`!`	Enter terse mode
`%`	Exit terse mode
`^`	Spell-check the rest of the line

In terse mode, Aspell will not print lines beginning with '`*`', which indicate correct words. This significantly improves running speed when the driving program is going to ignore correct words anyway.

In addition to the above commands which are designed for Ispell compatibility Aspell also supports its own extensions. All Aspell extensions follow the following format.

```
$$command [data]
```

Where *data* may or may not be required depending on the particular command. Aspell currently supports the following commands:

`cs option,value`	Change a configuration option.
`cr option`	Prints the value of a configuration option.
`pp`	Returns a list of all words in the current personal wordlist.
`ps`	Returns a list of all words in the current session dictionary.
`l`	Returns the current language name.

`ra` *`mis,cor`* Add the word pair to the replacement dictionary for later use. Returns nothing.

Anything returned is returned on its own line. All lists returned have the following format

`num of items: item1, item2, etc`

(Part of the preceding section was directly copied out of the Ispell manual)

6.3 Notes on Storing Replacement Pairs

The `store_repl` method and the `$$ra` should be used because Aspell is able to learn from users misspellings. For example on the first pass a user misspells *beginning* as *beging* so Aspell suggests:

```
begging, begin, being, Beijing, bagging, ....
```

However the user then tries *begning* and Aspell suggests

```
beginning, beaning, begging, ...
```

so the user selects *beginning*. However, later on in the document the user misspells it as *begng* (**not** *beging*). Normally Aspell will suggest.

```
began, begging, begin, begun, ...
```

However because it knows the user misspelled *beginning* as *beging* it will instead suggest:

```
beginning, began, begging, begin, begun ...
```

I myself often misspelled beginning (and still do) as something close to begging and two many times wind up writing sentences such as "begging with . . .".

Please also note that replacements commands has a memory. Which means if you first store the replacement pair:

```
sicolagest -> psycolagest
```

then store the replacement pair

```
psycolagest -> psychologist
```

The replacement pair

```
sicolagest -> psychologist
```

will also get stored so that you don't have to worry about it.

7 Adding Support For Other Languages

Before you consider adding support for Aspell, first make sure that someone else has not already done it. A good number of dictionaries off the Aspell home page at `http://aspell.net`. If your language is not listed above please send me a note and I will work with you on adding support.

Adding a language to Aspell is fairly straightforward. You basically need to create the language data file, and compile a new word list.

7.1 The Language Data File

The basic format of the language data file is the same as it is for the Aspell configuration file. It is named '*lang*`.dat`' and is located in the architecture independent data dir for Aspell (option '`data-dir`') which is usually '*prefix*`/share/aspell`'. Use `aspell config` to find out where it is in your installation. By convention the language name should be the two letter ISO 639 language code if it exists, if not use the three letter code.

The language data file has several mandatory fields, and several optional ones. All fields are case sensitive and should be in all lower case.

The two mandatory fields are '`name`' and '`charset`'.

'`name`' is the name of the language and should be the same as the file name (without the '`.dat`').

'`charset`' is the 8-bit character set Aspell will expect the word lists to be formatted in. If possible choose from one of the standard ones provided with Aspell. These are '`iso-8859-*`', '`koi8-*`', or '`viscii`'. If your language does not require any non-ascii characters choose '`iso-8859-1`'. If one of these standard character sets is not suitable for your language than you can create a new one. See Section 7.8 [Creating A New Character Set], page 41.

The optional fields are as follows:

'`data-encoding`'
: The encoding the language data files are expected to be in as well as the default encoding to use when saving the personal dictionaries. It can be either '`utf-8`' or any of the 8-bit encoding that Aspell supports. If not set, then it defaults to '`charset`'.

'`special`'
: Non-letter characters that can appear in your language such as the '`'`' and '`-`'. The format for the value is a list separated by spaces. Each item of the list has the following format.

```
<char> <begin><middle><end>
```

char is the non letter character in question. *begin*, *middle*, *end* are either a '`-`' or a '`*`'. A star for *begin* means that the character can begin a word, a '`-`' means it can't. The same is true for *middle* and *end*. For example, the entry for the '`'`' in English is:

```
' -*-
```

To include more than one middle character just list them one after another on the same line. For example, to make both the '`'`' and the '`-`' a middle character, use the following line in the language data file:

```
special ' -*- - -*-
```

‘`soundslike`’
: The name of the soundslike data for the language. The data is expected to be in the file ‘*name*`_phonetic.dat`’.

  If *name* is ‘`simpile`’ then a very simple soundslike is used. This is nearly as powerful as full phonetic soundslike but it can be computed a lot faster. (see Section 7.4 [The Simple Soundslike], page 37)

  If the soundslike name is ‘`none`’, or this option is not specified, than no soundslike will be used. The effective soundslike is the word converted to all lowercase and possibly with accents stripped depending on the ‘`store-as`’ option. For languages with phonetic spelling the difference will not be very noticeable. However, for languages with non-phonetic spelling there will be a noticeable difference. The difference you notice will depend on the quality of the soundslike data file. If you do not notice much of a difference for a language with non-phonetic spelling that is a good indication that the soundslike data is not rough enough—or the words you are trying are not that badly misspelled.

‘`invisible-soundslike`’
: Avoid storing the soundslike information with the word. Instead it is computed as needed. This option defaults to true if the soundslike is ‘`none`’ or ‘`simpile`’, and false when a phonetic soundslike is used.

‘`repl-table`’
: See Section 7.5 [Replacement Tables], page 38.

‘`keyboard`’
: The base name of the keyboard definition file to use. For more information see Section 4.4.3 [Notes on Typo-Analysis], page 19.

‘`sug-split-char`’
: A list of characters which specifies which characters to insert between two words when a word is split. This is a list option.

‘`affix`’
‘`affix-compress`’
‘`partially-expand`’
: See Section 7.6 [Affix Compression], page 38.

‘`store-as`’
: How the words are indexed in the dictionary. If "stripped" then the word is indexed in a lower case and de-accented form. If "lower", then the word is indexed in a lower case form but with accent info still intact. This just controls how the word is indexed, not how it is stored. The default is "stripped" unless affix compression is used.

‘`norm-required`’
: Should be set to true if your language makes use of private use characters or when Normalization Form C is not the same as full composition.

‘`normalize`’
‘`norm-form`’

Additional options includes options to control how run-together words are handled the same way as they are in the normal configuration files. for more information, please Section 7.7 [Controlling the Behavior of Run-together Words], page 40.

7.2 Compiling the Word List

Once you have a working language data file installed in the right place you are ready to compile the main word list. To find out what to do, see Chapter 5 [Working With Dictionaries], page 21. This section also includes instructions for creating the AWLI file.

7.3 Phonetic Code

Aspell is in fact the spell checker that comes up with the best suggestions if it finds an unknown word. One reason is that it does not just compare the word with other words in the dictionary (like Ispell does) but also uses phonetic comparisons with other words.

The new table driven phonetic code is very flexible and setting up phonetic transformation rules for other languages is not difficult but there can be a number of stumbling blocks — that's why I wrote this section.

The main phonetic code is free of any language specific code and should be powerful enough to allow setting up rules for any language. Anything which is language specific is kept in a plain text file and can easily be edited. So it's even possible to write phonetic transformation rules if you don't have any programming skills. All you need to know is how words of the language are written and how they are pronounced.

7.3.1 Syntax of the transformation array

In the translation array there are two strings on each line; the first one is the search string (or switch name) and the second one is the replacement string (or switch parameter). The line

```
version   version
```

is also required to appear somewhere in the translation array. The version string can be anything but it should be changed whenever a new version of the translation array is released. This is important because it will keep Aspell from using a compiled dictionary with the wrong set of rules. For example, if when coming up with suggestion for `hallo`, Aspell will use the new rules to come up with the soundslike say `H*L*`, but if ‘`hello`’ is stored in the dictionary using the old rules as `HL` instead of `H*L*` Aspell will never be able to come up with ‘`hello`’. So to solve this problem Aspell checks if the version strings match and aborts with an error if they don't. Thus it is important to update it whenever a new version of the translation array is released. This is only a problem with the main word list as the personal word lists are now stored as simple word lists with a single header line (i.e. no soundslike data).

Each non switch line represents one replacement (transformation) rule. Words beginning with the same letter must be grouped together; the order inside this group does not depend on alphabetical issues but it gives priorities; the higher the rule the higher the priority. That's why the first rule that matches is applied. In the following example:

```
GH   _
G    K
```

'GH -> _' has higher priority than 'G -> K'

'_' represents the empty string "". If 'GH -> _' came after 'G -> K', the second rule would never match because the algorithm would stop searching for more rules after the first match. The above rules transform any 'GH' to an empty string (delete them) and transforms any other 'G' to 'K'.

At the end of the first string of a line (the search string) there may optionally stand a number of characters in brackets. One (only one!) of these characters must fit. It's comparable with the '[ ]' brackets in regular expressions. The rule 'DG(EIY) -> J' for example would match any 'DGE', 'DGI' and 'DGY' and replace them with 'J'. This way you can reduce several rules to one.

Before the search string, one or more dashes '-' may be placed. Those search strings will be matched totally but only the beginning of the string will be replaced. Furthermore, for these rules no follow-up rule will be searched (what this is will be explained later). The rule 'TCH-- '-> _ will match any word containing 'TCH' (like 'match') but will only replace the first character 'T' with an empty string. The number of dashes determines how many characters from the end will not be replaced. After the replacement, the search for transformation rules continues with the not replaced 'CH'!

If a '<' is appended to the search string, the search for replacement rules will continue with the replacement string and not with the next character of the word. The rule 'PH< -> F' for example would replace 'PH' with 'F' and then again start to search for a replacement rule for 'F...'. If there would also be rules like 'FO '-> 'O' and 'F -> _' then words like 'PHOXYZ' would be transformed to 'OXYZ' and any occurrences of 'PH' that are not followed by an 'O' will be deleted like 'PHIXYZ -> IXYZ'. The second replacement however is not applied if the priority of this rule is lower than the priority of the first rule.

Priorities are added to a rule by putting a number between 0 and 9 at the end of the search string, for example 'ING6 -> N'. The higher the number the higher is the priority.

Priorities are especially important for the previously mentioned follow-up rules. Follow-up rules are searched beginning from the last string of the first search string. This is a bit complicated but I hope this example will make it clearer:

```
CHS      X
CH       G

HAU--1   H

SCH      SH
```

In this example 'CHS' in the word 'FUCHS' would be transformed to 'X'. If we take the word 'DURCHSCHNITT' then things look a bit different. Here 'CH' belongs together and 'SCH' belongs together and both are spoken separately. The algorithm however first finds the string 'CHS' which may not be transformed like in the previous word 'FUCHS'. At this point the algorithm can find a follow up rule. It takes the last character of the first matching rule ('CHS') which is 'S' and looks for the next match, beginning from this character. What it finds is clear: It finds 'SCH -> SH', which has the same priority (no priority means standard priority, which is 5). If the priority is the same or higher the follow-up rule will be applied.

Let's take a look at the word 'SCHAUKEL'. In this word 'SCH' belongs together and may not be taken apart. After the algorithm has found 'SCH '-> 'SH' it searches for a follow-up rule for 'H+''AUKEL'. It finds 'HAU--1 -> H', but does not apply it because its priority is lower than the one of the first rule. You see that this is a very powerful feature but it also can easily lead to mistakes. If you really don't need this feature you can turn it off by putting the line:

```
followup      0
```

at the beginning of the phonetic table file. As mentioned, for rules containing a '-' no follow-up rules are searched but giving such rules a priority is not totally senseless because they can be follow-up rules and in that case the priority makes sense again. Follow-up rules of follow-up rules are not searched because this is in fact not needed very often.

The control character '^' says that the search string only matches at the beginning of words so that the rule 'RH -> R' will only apply to words like 'RHESUS' but not 'PERHAPS'. You can append another '^' to the search string. In that case the algorithm treats the rest of the word totally separately from the first matched string at the beginning. This is useful for prefixes whose pronunciation does not depend on the rest of the word and vice versa like 'OVER^^' in English for example.

The same way as '^' works does '$' only apply to words that end with the search string. 'GN$ -> N' only matches on words like 'SIGN' but not 'SIGNUM'. If you use '^' and '$' together, both of them must fit 'ENOUGH^$ -> NF' will only match the word 'ENOUGH' and nothing else.

Of course you can combine all of the mentioned control characters but they must occur in this order: '< - priority ^ $'. All characters must be written in CAPITAL letters.

If absolutely no rule can be found — might happen if you use strange characters for which you don't have any replacement rule — the next character will simply be skipped and the search for replacement rules will continue with the rest of the word.

If you want double letters to be reduced to one you must set up a rule like 'LL- -> L'. If double letters in the resulting phonetic word should be allowed, you must place the line:

```
collapse_result     0
```

at the beginning of your transformation table file; otherwise set the value to '1'. The English rules for example strip all vowels from words and so the word "GOGO" would be transformed to "K" and not to "KK" (as desired) if `collapse_result` is set to 1. That's why the English rules have `collapse_result` set to 0.

By default, all accents are removed from a word before it is matched to the soundslike rules. If you do not want this then add the line

```
remove_accents      0
```

at the beginning of your file. The exact definition of an accent is language dependent and is controlled via the character set file. If you set remove_accents to '0' then you should also set "store-as" to "lower" in the language data file (not the phonetic transformation file) otherwise Aspell will have problems when both the accented and the de-accented version of a word appearing in the dictionary; it will consider one of them as incorrectly spelled.

7.3.2 How do I start finally?

Before you start to write an array of transformation rules, you should be aware that you have to do some work to make sure that things you do will result in correct transformation rules.

7.3.2.1 Things that come in handy

First of all, you need to have a large word list of the language you want to make phonetics for. It should contain about as many words as the dictionary of the spell checker. If you don't have such a list, you will probably find an Ispell dictionary at `http://fmg-www.cs.ucla.edu/geoff/ispell-dictionaries.html` which will help you. You can then make affix expansion via `ispell -e` and then pipe it through `tr " " "\n"` to put one word on each line. After that you eventually have to convert special characters like '`é`' from Ispell's internal representation to latin1 encoding. `sed s/e'/é/g` for example would replace all '`e'`' with '`é`'.

The second is that you know how to use regular expressions and know how to use `grep`. You should for example know that:

```
grep ^[^aeiou]qu[io] wordlist | less
```

will show you all words that begin with any character but '`a`', '`e`', '`i`', '`o`' or '`u`' and then continue with '`qui`' or '`quo`'. This stuff is important for example to find out if a phonetic replacement rule you want to set up is valid for all words which match the expression you want to replace. Taking a look at the regex(7) man page is a good idea.

7.3.2.2 What the phonetic code should do

Normal text comparison works well as long as the typer misspells a word because he pressed one key he didn't really want to press. In these cases, mostly one character differs from the original word.

In cases where the writer didn't know about the correct spelling of the word, the word may have several characters that differ from the original word but usually the word would still sound like the original. Someone might think that 'tough' is spelled 'taff'. No spell checker without phonetic code will come to the idea that this might be 'tough', but a spell checker who knows that 'taff' would be pronounced like 'tough' will make good suggestions to the user. Another example could be 'funetik' and 'phonetic'.

From these examples you can see that the phonetic transformation should not be too fussy and too precise. If you implement a whole phonetic dictionary as you can find it in books this will not be very useful because then there could still be many characters differing from the misspelled and the desired word. What you should do if you implement the phonetic transformation table is to reduce the number of used letters to the only really necessary ones.

Characters that sound similar should be reduced to one. In English language for example 'Z' sounds like 'S' and that's why the transformation rule '`Z -> S`' is present in the replacement table. "PH is spoken like "F and so we have a '`PH -> F`' rule.

If you take a closer look you will even see that vowels sound very similar in the English language: 'contradiction', 'cuntradiction', 'cantradiction' or 'centradiction' in fact sound nearly the same, don't they? Therefore the English phonetic replacement rules not only reduce all vowels to one but even remove them all (removing is done by just setting up no rule for those letters). The phonetic code of "contradiction" is "KNTRTKXN" and if you try to read this letter-monster loud you will hear that it still sound a bit like 'contradiction'. You also see that "D" is transformed to "T" because they nearly sound the same.

If you think you have found a regularity you should *always* take your word list and `grep` for the corresponding regular expression you want to make a transformation rule for. An

example: If you come to the idea that all English words ending on 'ough' sound like 'AF' at the end because you think of 'enough' and 'tough'. If you then `grep` for the corresponding regular expression by `grep -i ough$ wordlist` you will see that the rule you wanted to set up is not correct because the rule doesn't fit to words like 'although' or 'bough'. So you have to define your rule more precisely or you have to set up exceptions if the number of words that differ from the desired rule is not too big.

Don't forget about follow-up rules which can help in many cases but which also can lead to confusion and unwanted side effects. It's also important to write exceptions in front of the more general rules ('`GH`' before '`G`' etc.).

If you think you have set up a number of rules that may produce some good results try them out! If you run Aspell as `aspell --lang=`*`your_language`* `pipe` you get a prompt at which you can type in words. If you just type words Aspell checks them and eventually makes suggestions if they are misspelled. If you type in `$$Sw` *`word`* you will see the phonetic transformation and you can test out if your work does what you want.

Another good way to check that changes you make to your rules don't have any bad side effects is to create another list from your word list which contains not only the word of the word list but also the corresponding phonetic version of this word on the same line. If you do this once before the change and once after the change you can make a diff (see `man diff`) to see what *really* changed. To do this use the command `aspell --lang=`*`your_language`* `soundslike`. In this mode aspell will output the the original word and then its soundslike separated by a tab character for each word you give it. If you are interested in seeing how the algorithm works you can download a set of useful programs from `http://members.xoom.com/maccy/spell/phonet-utils.tar.gz`. This includes a program that produces a list as mentioned above and another program which illustrates how the algorithm works. It uses the same transformation table as Aspell and so it helps a lot during the process of creating a phonetic transformation table for Aspell.

During your work you should write down your basic ideas so that other people are able to understand what you did (and you still know about it after a few weeks). The English table has a huge documentation appended as an example.

Now you can start experimenting with all the things you just read and perhaps set up a nice phonetic transformation table for your language to help Aspell to come up with the best correction suggestions ever seen also for your language. Take a look at the Aspell homepage to see if there is already a transformation table for your language. If there is one you might also take a look at it to see if it could be improved.

If you think that this section helped you or if you think that this is just a waste of time you can send any feedback to `bjoern.jacke@gmx.de`.

7.4 The Simple Soundslike

The simple soundslike goes something like this:

```
sl0[0] = lookup0(word[0])
for (i = 1; i < size; i++)
  sl0[i] = lookup(word[i]);
s = 0;
for (i = 0; i < size; i++)
  sl.append(al0[i]) unless sl0[i] == 0 || sl0[i] == sl0[i-1];
```

Basically each character can be converted to another character or deleted. A separate lookup table is used for the first character. If the same soundslike letter is repeated, the duplicate is removed.

By default all accents are removed, and all vowels are deleted unless they appear at the start of the word in which case they are converted to a '`*`'. The exact behavior can be customized via the character data file.

The simplified soundslike has the advantage that it is very fast to compute and thus does not need to be stored with a word. Also, when affix compression is used and the '`partially-expand`' is given the results will be identical to the results when affix compression is not used.

Of course it is not nearly as powerful as the phonetic soundslike.

7.5 Replacement Tables

When phonetic code is not used a replacement table can be used instead. To enable the use of a replacement table add the line `repl-table` *`lang`*, in which case the replacement table is excepted to be in the file '*`lang`*`_repl.dat`'. A complete file name can also be specified in place of *lang*. For compatibility with MySpell the replacement table can also be part of the affix file, in which case '`repl-table`' will be '*`lang`*`_affix.dat"`'.

Replacement table syntax:

```
REP [number_of_replacement_definitions]
REP [what] [replacement]
REP [what] [replacement]
```

For example a possible English replacement table definition to handle misspelled consonants:

```
REP 8
REP f ph
REP ph f
REP f gh
REP gh f
REP j dg
REP dg j
REP k ch
REP ch k
```

7.6 Affix Compression

Aspell, as of version 0.60, now has support for affix compression. The codebase comes from MySpell found in OpenOffice.

To add support for affix compression add the following lines to the language data file.

```
affix          lang
affix-compress true
```

The line '`affix` *`lang`*' adds support for recognizing affix information, and the line '`affix-compress true`' enables affix compression.

The affix file is expected to be named '*lang*_affix.dat'. It is the exact same format as those used by MySpell. More information can be found in the myspell/ directory of the distribution or at http://lingucomponent.openoffice.org/dictionary.html.

Affix compression can also be used with soundslike lookup. Aspell does this by only storing the soundslike for the root word. When a word is misspelled it will search for a soundslike close to all possible roots of the misspelled word.

When no soundslike information, or the simple soundslike, is used it may be beneficial to specify the option 'partially-expand' which will partially expand a word with affix information so that the affix flags do not effect the first 3 letters of the word. This will allow Aspell to get more accurate results when scanning the list for near misses since the full word can be used and not just the root. Specifying this option, however, will also effectively expand any prefixes. Thus this option should not be used for prefix heavy languages such as Hebrew.

An existing word list, without affix info, can be affix compressed using using `aspell munch-list`.

7.6.1 Format of the Affix File

An affix is either a prefix or a suffix attached to root words to make other words. For example supply -> supplied by dropping the "y" and adding an "ied" (the suffix).

Here is an example of how to define one specific suffix borrowed from the English affix file.

```
SFX D Y 4
SFX D   0     d          e
SFX D   y     ied        [^aeiou]y
SFX D   0     ed         [^ey]
SFX D   0     ed         [aeiou]y
```

This file is space delimited and case sensitive. So this information can be interpreted as follows:

The first line has 4 fields:

1	SFX	indicates this is a suffix
2	D	is the name of the character which represents this suffix
3	Y	indicates it can be combined with prefixes (cross product)
4	4	indicates that sequence of 4 affix entries are needed to properly store the affix information

The remaining lines describe the unique information for the 4 affix entries that make up this affix. Each line can be interpreted as follows: (note fields 1 and 2 are used as a check against line 1 info)

1	SFX	indicates this is a suffix
2	D	is the name of the character which represents this affix
3	y	the string of chars to strip off before adding affix (a 0 here indicates the NULL string)
4	ied	the string of affix characters to add (a 0 here indicates the NULL string)
5	[^aeiou]y	the conditions which must be met before the affix can be applied

Field 5 is interesting. Since this is a suffix, field 5 tells us that there are 2 conditions that must be met. The first condition is that the next to the last character in the word

must *not* be any of the following "a", "e", "i", "o" or "u". The second condition is that the last character of the word must end in "y".

7.6.2 When Compared With Ispell

Now for comparison purposes, here is the same information from the Ispell '`english.aff`' compression file which was used as the basis for the OOo one.

```
flag *D:
    E           >       D               # As in create > created
    [^AEIOU]Y   >       -Y,IED          # As in imply > implied
    [^EY]       >       ED              # As in cross > crossed
    [AEIOU]Y    >       ED              # As in convey > conveyed
```

The Ispell information has exactly the same information but in a slightly different (case-insensitive) format:

Here are the ways to see the mapping from Ispell .aff format to our OOo format.

1. The ispell english.aff has flag D under the "suffix" section so you know it is a suffix.
2. The D is the character assigned to this suffix
3. '`*`' indicates that it can be combined with prefixes
4. Each line following the : describes the affix entries needed to define this suffix
   - The first field is the conditions that must be met.
   - The second field is after the `>` if a `"-"` occurs is the string to strip off (can be blank).
   - The third field is the string to add (the affix)

In addition all chars in ispell aff files are in uppercase.

7.6.3 Specifying Affix Flags

Affix flags are specified in the word list by specifying them after the '`/`' character:

```
word/flags
```

For example:

```
create/DG
```

will associate the '`D`' and '`G`' flag with the word create.

7.7 Controlling the Behavior of Run-together Words

Aspell currently has support for unconditionally accepting run-together words.

Support for unconditionally accepting run-together words can either be turned on in the language data file or as a normal option via the '`run-together`' option. The '`run-together-limit`' options controls the maximum number of words that can be strung together, the default is normally 2. The '`run-together-min`' options controls the minimum length of the individual components of the run together word, the default is normally 3. Both the '`run-together-limit`' and '`run-together-min`' option may be specified in both the language data file or as a normal option.

7.8 Creating A New Character Set

If there is not a standard character set for your language than you can invent one. The new charset will only be used by Aspell internally. If the option '`data-encoding`' is set to '`utf-8`', and your current locale character type is always set to '`utf-8`', than you can use UTF-8 for everything and not worry yourself that an 8-bit character set is being used internally. If your language has no more than 210 distinct symbols, including different capitalizations and accents, than Aspell can support it.

In order to create a new character set you will need to download the file '`aspell-lang-`*`version`*`.tar.bz2`' from `ftp://ftp.gnu.org/gnu/aspell/` and follow the instructions in the '`README`' file. Version 0.1, and 0.2 of mkchardata *will not* work as the format of the character data file has changed.

7.9 Format of the Character Set Data File

The character set data files that Aspell uses can be fine tuned. If this is done, that character set file should be renamed to '*`lang`*`.cset`'.

The line(s) starting with '`=`' do not specify the exact name of the file, but rather the name of the Unicode mapping that is used. Unless you change the Unicode mapping, this line should not be changed.

7.10 Creating An Official Dictionary Package

Once you have a basic dictionary working, you should consider creating an official package so that it can be distributed with Aspell. To do so download `ftp://alpha.gnu.org/gnu/aspell/aspell-lang-`*`version`*`.tar.bz2`.

Appendix A Implementation Notes

A.1 Aspell Suggestion Strategy

The magic behind my spell checker comes from merging Lawrence Philips excellent metaphone algorithm and Ispell's near miss strategy which is inserting a space or hyphen, interchanging two adjacent letters, changing one letter, deleting a letter, or adding a letter.

The process goes something like this.

1. Convert the misspelled word to its soundslike equivalent (its metaphone for English words).
2. Find all words that have a soundslike within one or two edit distances from the original word's soundslike. The edit distance is the total number of deletions, insertions, exchanges, or adjacent swaps needed to make one string equivalent to the other. When set to only look for soundslikes within one edit distance it tries all possible soundslike combinations and checks if each one is in the dictionary. When set to find all soundslike within two edit distances it scans through the entire dictionary and quickly scores each soundslike. The scoring is quick because it will give up if the two soundslikes are more than two edit distances apart.
3. Find misspelled words that have a correctly spelled replacement by the same criteria of step number 2 and 3. That is the misspelled word in the word pair (such as "teh -> the") would appear in the suggestions list as if it was a correct spelling.
4. Score the result list and return the words with the lowest score. The score is roughly the weighed average of the weighed edit distance of the word to the misspelled word and the soundslike equivalent of the two words. The weighted edit distance is like the edit distance except that the various edits have weights attached to them.
5. Replace the misspelled words that have correctly spelled replacements with their replacements and remove any duplicates that might arise because of this.

Please note that the soundslike equivalent is a rough approximation of how the words sounds. It is not the phoneme of the word by any means. For more details about exactly how each step is performed please see the file '`suggest.cc`'. For more information on the metaphone algorithm please see the data file '`english_phonet.dat`'.

A.2 Notes on 8-bit Characters

There is a very good reason I use 8-bit characters in Aspell. Speed and simplicity. While many parts of my code can fairly easily be converted to some sort of wide character as my code is clean. Other parts can not be.

One of the reasons because is many, many places I use a direct lookup to find out various information about characters. With 8-bit characters this is very feasible because there is only 256 of them. With 16-bit wide characters this will waste a LOT of space. With 32-bit characters this is just plain impossible. Converting the lookup tables to another form is certainly possible but degrades performance significantly.

Furthermore, some of my algorithms relay on words consisting only on a small number of distinct characters (often around 30 when case and accents are not considered). When the possible character can consist of any Unicode character this number becomes several

thousand, if that. In order for these algorithms to still be used, some sort of limit will need to be placed on the possible characters the word can contain. If I impose that limit, I might as well use some sort of 8-bit characters set which will automatically place the limit on what the characters can be.

There is also the issue of how I should store the word lists in memory? As a string of 32 bit wide characters. Now that is using up 4 times more memory than characters would and for languages that can fit within an 8-bit character that is, in my view, a gross waste of memory. So maybe I should store them is some variable width format such as UTF-8. Unfortunately, way, way too many of the algorithms will simply not work with variable width characters without significant modification which will very likely degrade performance. So the solution is to work with the characters as 32-bit wide characters and than convert it to a shorter representation when storing them in the lookup tables. Now that can lead to an inefficiency. I could also use 16 bit wide characters however that may not be good enough to hold all future versions of Unicode and therefore has the same problems.

As a response to the space waste used by storing word lists in some sort of wide format some one asked:

> Since hard drives are cheaper and cheaper, you could store a dictionary in a usable (uncompressed) form and use it directly with memory mapping. Then the efficiency would directly depend on the disk caching method, and only the used part of the dictionaries would really be loaded into memory. You would no more have to load plain dictionaries into main memory, you'll just want to compute some indexes (or something like that) after mapping.

However, the fact of the matter is that most of the dictionary will be read into memory anyway if it is available. If it is not available than there would be a good deal of disk swaps. Making characters 32-bit wide will increase the chance that there are more disk swaps. So the bottom line is that it is more efficient to convert characters from something like UTF-8 into some sort of 8-bit character. I could also use some sort of disk space lookup table such as the Berkeley Database. However this will **definitely** degrade performance.

The bottom line is that keeping Aspell 8-bit internally is a very well though out decision that is not likely to change any time soon. Feel free to challenge me on it, but, don't expect me to change my mind unless you can bring up some point that I have not thought of before and quite possibly a patch to solve cleanly convert Aspell to Unicode internally without a serious performance lost OR serious memory usage increase.

Appendix B Languages Which Aspell can Support

Even though Aspell will remain 8-bit internally it should still be able to support any written languages not based on a logographic script. The only logographic writing system in current use are those based on hànzi which includes Chinese, Japanese, and sometimes Korean.

B.1 Supported

Aspell 0.60 should be able to support the following languages:

Code	Language Name	Script	Dictionary Available	Gettext Translation
aa	Afar	Latin	-	-
af	Afrikaans	Latin	0.50	-
ak	Akan	Latin	Planned	-
am	Amharic	Ethiopic	0.60	-
ar	Arabic	Arabic	Planned	-
as	Assamese	Bengali	-	-
ast	Asturian / Bable	Latin	Maybe	-
av	Avar	Cyrillic	-	-
ay	Aymara	Latin	-	-
az	Azerbaijani	Latin	0.60	-
ba	Bashkir	Cyrillic	-	-
ban	Balinese	Latin	Maybe	-
be	Belarusian	Cyrillic	0.50	Incomplete
bem	Bemba	Latin	Maybe	-
bg	Bulgarian	Cyrillic	0.50	-
bh	Bihari	Devanagari	-	-
bi	Bislama	Latin	Maybe	-
bm	Bambara	Latin	-	-
bn	Bengali	Bengali	0.60	-
bo	Tibetan	Tibetan	-	-
br	Breton	Latin	0.50	-
bs	Bosnian	Latin	Maybe	-
ca	Catalan / Valencian	Latin	0.50	-
ce	Chechen	Cyrillic	-	-
ceb	Cebuano	Latin	Maybe	-
ch	Chamorro	Latin	Maybe	-
chk	Chuukese	Latin	Maybe	-
co	Corsican	Latin	Maybe	-
cs	Czech	Latin	0.50	Yes
csb	Kashubian	Latin	0.60	-
cv	Chuvash	Cyrillic	-	-
cy	Welsh	Latin	0.50	-
da	Danish	Latin	0.50	Incomplete
de	German	Latin	0.50	Yes
ee	Ewe	Latin	-	-
el	Greek	Greek	0.50	-

en	English	Latin	0.50	Yes
eo	Esperanto	Latin	0.50	-
es	Spanish	Latin	0.50	Incomplete
et	Estonian	Latin	0.60	-
eu	Basque	Latin	Maybe	-
fa	Persian	Arabic	0.60	-
ff	Fulah	Latin	-	-
fi	Finnish	Latin	0.60	-
fj	Fijian	Latin	Maybe	-
fo	Faroese	Latin	0.50	-
fr	French	Latin	0.50	Yes
fur	Friulian	Latin	Maybe	-
fy	Frisian	Latin	Maybe	-
ga	Irish	Latin	0.50	Yes
gd	Scottish Gaelic	Latin	0.50	-
gl	Gallegan	Latin	0.50	-
gn	Guarani	Latin	Maybe	-
gu	Gujarati	Gujarati	Maybe	-
gv	Manx Gaelic	Latin	0.50	-
ha	Hausa	Latin	Maybe	-
haw	Hawaiian	Latin	Maybe	-
he	Hebrew	Hebrew	0.60	-
hi	Hindi	Devanagari	0.60	-
hil	Hiligaynon	Latin	0.50	-
ho	Hiri Motu	Latin	-	-
hr	Croatian	Latin	0.50	-
hsb	Upper Sorbian	Latin	0.60	-
ht	Haitian Creole	Latin	Maybe	-
hu	Hungarian	Latin	0.60	-
hy	Armenian	Armenian	-	-
hz	Herero	Latin	-	-
ia	Interlingua (IALA)	Latin	0.50	-
iba	Iban	Latin	Maybe	-
id	Indonesian	Latin	0.50	-
ig	Igbo	Latin	Maybe	-
ii	Sichuan Yi	Yi	-	-
ilo	Iloko	Latin	Maybe	-
io	Ido	Latin	-	-
is	Icelandic	Latin	0.50	-
it	Italian	Latin	0.50	Yes
jv	Javanese	Javanese, Latin	Maybe	-
ka	Georgian	Georgian	-	-
kac	Kachin	Latin	Maybe	-
kg	Kongo	Latin	Maybe	-
kha	Khasi	Bengali, Latin, Latin	Maybe	-
ki	Kikuyu / Gikuyu	Latin	-	-
kj	Kwanyama	Latin	-	-

kk	Kazakh	Cyrillic	-	-
kl	Kalaallisut / Greenlandic	Latin	Maybe	-
kn	Kannada	Kannada	-	-
kok	Konkani	Latin	Maybe	-
kr	Kanuri	Latin	-	-
ks	Kashmiri	Arabic	-	-
ku	Kurdish	Arabic, Cyrillic, Latin	0.50	-
kv	Komi	Cyrillic	-	-
kw	Cornish	Latin	Maybe	-
ky	Kirghiz	Cyrillic	-	-
la	Latin	Latin	0.60	-
lb	Luxembourgish	Latin	Maybe	-
lg	Ganda	Latin	Maybe	-
li	Limburgian	Latin	Maybe	-
ln	Lingala	Latin	Maybe	-
loz	Lozi	Latin	Maybe	-
lt	Lithuanian	Latin	0.60	-
lu	Luba-Katanga	Latin	-	-
luo	Luo (Kenya and Tanzania)	Latin	Maybe	-
lv	Latvian	Latin	0.60	-
mg	Malagasy	Latin	0.50	-
mh	Marshallese	Latin	Maybe	-
mi	Maori	Latin	0.50	-
min	Minangkabau	Latin	Maybe	-
mk	Macedonian	Cyrillic	0.50	-
ml	Malayalam	Malayalam	Planned	-
mn	Mongolian	Cyrillic	0.60	Yes
mo	Moldavian	Cyrillic	-	-
mr	Marathi	Devanagari	0.60	-
ms	Malay	Latin	0.50	-
mt	Maltese	Latin	0.50	-
my	Burmese	Myanmar	-	-
nb	Norwegian Bokmal	Latin	0.50	-
nd	North Ndebele	Latin	Maybe	-
nds	Low Saxon	Latin	0.60	-
ne	Nepali	Devanagari	Maybe	-
ng	Ndonga	Latin	Maybe	-
niu	Niuean	Latin	Maybe	-
nl	Dutch	Latin	0.50	Yes
nn	Norwegian Nynorsk	Latin	0.50	-
nr	South Ndebele	Latin	-	-
nso	Northern Sotho	Latin	Maybe	-
nv	Navajo	Latin	-	-
ny	Nyanja	Latin	0.50	-
oc	Occitan / Provencal	Latin	Maybe	-

om	Oromo	Latin	-	-
or	Oriya	Oriya	0.60	-
os	Ossetic	Cyrillic	-	-
pa	Punjabi	Gurmukhi	0.60	-
pam	Pampanga	Latin	Maybe	-
pap	Papiamento	Latin	Maybe	-
pl	Polish	Latin	0.50	-
ps	Pushto	Arabic	-	-
pt	Portuguese	Latin	0.50	Incomplete
qu	Quechua	Latin	0.60	-
rar	Rarotongan	Latin	Maybe	-
rn	Rundi	Latin	Maybe	-
ro	Romanian	Latin	0.50	Incomplete
ru	Russian	Cyrillic	0.50	Yes
rw	Kinyarwanda	Latin	0.50	-
sc	Sardinian	Latin	0.50	-
sd	Sindhi	Arabic	-	-
se	Northern Sami	Latin	Maybe	-
sg	Sango	Latin	Maybe	-
si	Sinhalese	Sinhala	-	-
sk	Slovak	Latin	0.50	-
sl	Slovenian	Latin	0.50	Yes
sm	Samoan	Latin	Maybe	-
sn	Shona	Latin	Maybe	-
so	Somali	Latin	Maybe	-
sq	Albanian	Latin	Maybe	-
sr	Serbian	Cyrillic, Latin	Planned	Incomplete
ss	Swati	Latin	Maybe	-
st	Southern Sotho	Latin	Maybe	-
su	Sundanese	Latin	Maybe	-
sv	Swedish	Latin	0.50	-
sw	Swahili	Latin	0.50	-
ta	Tamil	Tamil	0.60	-
te	Telugu	Telugu	0.60	-
tet	Tetum	Latin	0.50	-
tg	Tajik	Cyrillic	Maybe	Incomplete
ti	Tigrinya	Ethiopic	Maybe	-
tk	Turkmen	Latin	Planned	-
tkl	Tokelau	Latin	Maybe	-
tl	Tagalog	Latin	0.50	-
tn	Tswana	Latin	0.50	-
to	Tonga	Latin	Maybe	-
tpi	Tok Pisin	Latin	Maybe	-
tr	Turkish	Latin	0.50	-
ts	Tsonga	Latin	Maybe	-
tt	Tatar	Cyrillic	-	-
tw	Twi	Latin	-	-

ty	Tahitian	Latin	Maybe	-
ug	Uighur	Arabic	-	-
uk	Ukrainian	Cyrillic	0.50	Yes
ur	Urdu	Arabic	Maybe	-
uz	Uzbek	Cyrillic	0.60	-
ve	Venda	Latin	Maybe	-
vi	Vietnamese	Latin	0.60	Yes
wa	Walloon	Latin	0.50	Incomplete
wo	Wolof	Latin	Maybe	-
xh	Xhosa	Latin	Maybe	-
yi	Yiddish	Hebrew	0.60	-
yo	Yoruba	Latin	Maybe	-
za	Zhuang	Latin	-	-
zu	Zulu	Latin	0.50	-

Dictionaries marked as *0.50* are available for Aspell 0.50. Ones marked as *0.60* are available for Aspell 0.60 only. Ones marked as *Planned* should eventually be available. Ones marked as *Maybe* might be available in the future. See Section B.4 [Planned Dictionaries], page 51, for more info.

B.1.1 Notes on Latin Languages

Any word that can be written using one of the Latin ISO-8859 character sets (ISO-8859-1,2,3,4,9,10,13,14,15,16) can be written, in decomposed form, using the ASCII characters, the 23 additional letters:

```
U+00C6 LATIN CAPITAL LETTER AE
U+00D0 LATIN CAPITAL LETTER ETH
U+00D8 LATIN CAPITAL LETTER O WITH STROKE
U+00DE LATIN CAPITAL LETTER THORN
U+00DE LATIN SMALL LETTER THORN
U+00DF LATIN SMALL LETTER SHARP S
U+00E6 LATIN SMALL LETTER AE
U+00F0 LATIN SMALL LETTER ETH
U+00F8 LATIN SMALL LETTER O WITH STROKE
U+0110 LATIN CAPITAL LETTER D WITH STROKE
U+0111 LATIN SMALL LETTER D WITH STROKE
U+0126 LATIN CAPITAL LETTER H WITH STROKE
U+0127 LATIN SMALL LETTER H WITH STROKE
U+0131 LATIN SMALL LETTER DOTLESS I
U+0138 LATIN SMALL LETTER KRA
U+0141 LATIN CAPITAL LETTER L WITH STROKE
U+0142 LATIN SMALL LETTER L WITH STROKE
U+014A LATIN CAPITAL LETTER ENG
U+014B LATIN SMALL LETTER ENG
U+0152 LATIN CAPITAL LIGATURE OE
U+0153 LATIN SMALL LIGATURE OE
U+0166 LATIN CAPITAL LETTER T WITH STROKE
```

```
U+0167 LATIN SMALL LETTER T WITH STROKE
```

and the 14 modifiers:

```
U+0300 COMBINING GRAVE ACCENT
U+0301 COMBINING ACUTE ACCENT
U+0302 COMBINING CIRCUMFLEX ACCENT
U+0303 COMBINING TILDE
U+0304 COMBINING MACRON
U+0306 COMBINING BREVE
U+0307 COMBINING DOT ABOVE
U+0308 COMBINING DIAERESIS
U+030A COMBINING RING ABOVE
U+030B COMBINING DOUBLE ACUTE ACCENT
U+030C COMBINING CARON
U+0326 COMBINING COMMA BELOW
U+0327 COMBINING CEDILLA
U+0328 COMBINING OGONEK
```

Which is a total of 37 additional Unicode code points.

All ISO-8859 character leaves the characters 0x00 - 0x1F, and 0x80 - 0x9F unmapped as they are generally used as control characters. Of those, 0x01 - 0x0F, 0x11 - 0x1F and 0x80 - 0x9F may be mapped to anything in Aspell. This is a total of 62 characters which can be remapped in any ISO-8859 character set. Thus, by remapping 37 of the 62 characters to the previously specified Unicode code-points, any modified ISO-8859 character set can be used for any Latin languages covered by ISO-8859. Of course decomposing every single accented character wastes a lot of space, so only characters that cannot be represented in the precomposed form should be broken up. By using this trick it is possible to store foreign words in the correctly accented form in the dictionary even if the precomposed character is not in the current character set.

Any letter in the Unicode range U+0000 - U+0249, U+1E00 - U+1EFF (Basic Latin, Latin-1 Supplement, Latin Extended-A, Latin Extended-B, and Latin Extended Additional) can be represented using around 175 basic letters, and 25 modifiers which is less than 210 and can thus fit in an Aspell 8-bit character set. Since this Unicode range covers any possible Latin language this special character set can be used to represent any word written using the Latin script if so desired.

B.1.2 Syllabic

Syllabic languages use a separate symbol for each syllable of the language. Even thought most of them have more than 210 distinct symbols Aspell can still support them by breaking them up.

B.1.2.1 The Ethiopic Syllabary

Even though the Ethiopic script has more than 210 distinct characters Aspell can still handle it. The idea is to split each character into two parts based on the Consonant and Vowel parts. This encoding of the syllabary is far more useful to Aspell than if they were stored in UTF-8 or UTF-16. In fact, the exiting suggestion strategy of Aspell will work well with this encoding without any additional modifications. However, additional improvements may be possible by taking advantage of the consonant-vowel structure of this encoding.

In fact, the split consonant-vowel representation may prove to be so useful that it may be beneficial to encode other syllabary in this fashion, even if they are less than 210 of them.

The code to break up a syllabary into the consonant-vowel part is part of the Unicode normalization process.

B.1.2.2 The Yi Syllabary

A very large syllabary with 819 distinct symbols. However, like Ethiopic, it should be possible to support this script by breaking it up.

B.1.2.3 The Ojibwe Syllabary

With only 120 distinct symbols, Aspell can actually support this one as is. However, as previously mentioned, it may be beneficial to break it up into the consonant-vowel representation anyway.

B.2 Unsupported

These languages, when written in the given script, are currently unsupported by Aspell for one reason or another.

Code	Language Name	Script
ja	Japanese	Japanese
km	Khmer	Khmer
ko	Korean	Han, Hangul
lo	Lao	Lao
th	Thai	Thai
zh	Chinese	Han

B.2.1 The Thai, Khmer, and Lao Scripts

The Thai, Khmer, and Lao scripts presents a different problem for Aspell. The problem is not that there are more than 210 unique symbols, but that there are no spaces between words. This means that there is no easy way to split a sentence into individual words. However, it is still possible to spell check these scripts, it is just a lot more difficult. I will be happy to work with someone who is interested in adding Thai, Khmer, or Lao support to Aspell, but it is not likely something I will do on my own in the foreseeable future.

B.2.2 Languages which use Hànzi Characters

Hànzi Characters are used to write Chinese, Japanese, Korean, and were once used to write Vietnamese. Each hànzi character represents a syllable of a spoken word and also has a meaning. Since there are around 3,000 of them in common usage it is unlikely that Aspell will ever be able to support spell checking languages written using hànzi until full Unicode support is implemented. However, I am not even sure if these languages need spell checking since hànzi characters are generally not entered in directly. Furthermore even if Aspell could spell check hànzi the existing suggestion strategy will not work well at all, and thus a completely new strategy will need to be developed. However, if it is the case that hànzi needs to be spell checked and you know something about the issues involved please fell free to contact me.

B.2.3 Japanese

Modern Japanese is written in a mixture of *hiragana*, *katakana*, *kanji*, and sometimes *romaji*. *Hiragana* and *katakana* are both syllabaries unique to Japan, *kanji* is a modified form of hànzi, and *romaji* uses the Latin alphabet. With some work, Aspell should be able to check the non-kanji part of Japanese text. However, based on my limited understanding of Japanese hiragana is often used at the end of kanji. Thus if Aspell was to simply separate out the hiragana from kanji it would end up with a lot of word endings which are not proper words and will thus be flagged as misspellings. However, this can be fairly easily rectified as text is tokenized into words before it is converted into Aspell's internal encoding. In fact, some Japanese text is written in entirely in one script. For example books for children and foreigners are sometimes written entirely in hiragana. Thus, Aspell, in its current state, could prove at least somewhat useful for spell checking Japanese.

B.2.4 Hangul

Korean is generally written in hangul or a mixture of han and hangul. In Hangul letters individual letters, known as jamo, are grouped together in syllable blocks. Unicode allows Hangul to be stored in one of three ways, (A) Individual jamo letters (Hangul Compatibility Jamo, U+3130 - U+318F), (D) decomposed jamo (Hangul Jamo, U+1100 - U+11FF), and (C) precoposed sylable blocks (Hangul Syllables, U+AC00 - U+D7AF). In order for Aspell to work with Hangul it needs to be form A. Unfortunately the existing Normalization code in Aspell will not be able to adequately deal with converting Hangul from form D and C to form A and back again. However, once this code is written, Aspell should be able to spell check Hangul with out any problem.

B.3 Languages Written in Multiple Scripts

Aspell should be able to check text written in the same language but in multiple scripts with some work. If the number of unique symbols in both scripts is less than 210, then a special character set can be used to allow both scripts to be encoded in the same dictionary. However this may not be the most efficient solution. An alternate solution is to store each script in its own dictionary and allow Aspell to choose the correct dictionary based on which script the given word is written in. Aspell currently does not support this mode of spell checking but it is something that I hope to eventually support.

B.4 Notes on Planned Dictionaries

Kevin Patrick Scannell has word lists available for the following languages based on his web crawling software (`http://borel.slu.edu/crubadan/`) but needs someone to proofread them:

- Asturian (ast)
- Bali (ban)
- Bemba (bem)
- Bislama (bi)
- Cebuano (ceb)
- Chamorro (ch)
- Chuukese (chk)

- Corsican (co)
- Basque (eu)
- Fijian (fj)
- Friulian (fur)
- Frisian (fy)
- Guarani (gn)
- Hausa (ha)
- Hawaiian (haw)
- Hiligaynon (hil)
- Haitian Creole (ht)
- Iban (iba)
- Igbo (ig)
- Ilocano (ilo)
- Javanese (jv)
- Jingpho (kac)
- Kalaallisut (kl)
- Konkani (kok)
- Cornish (kw)
- Luganda (lg)
- Limburgish (li)
- Lingala (ln)
- Lozi (loz)
- Luo (luo)
- Marshallese (mh)
- Minangkabau (min)
- Ndebele (nd)
- Low Saxon (nds)
- Ndonga (ng)
- Niue (niu)
- Northern Sotho (nso)
- Occitan (oc)
- Pampangan (pam)
- Papiamentu (pap)
- Rundi (rn)
- Northern Saami (se)
- Samoan (sm)
- Shona (sn)
- Somali (so)
- Swati (ss)

- Southern Sotho (st)
- Sunda (su)
- Tajiki (tg)
- Tokelauan (tkl)
- Tongan (to)
- Tok Pisin (tpi)
- Tsonga (ts)
- Tahitian (ty)
- Venda (ve)
- Wolof (wo)
- Xhosa (xh)
- Yoruba (yo)

If you are interested, please contact him at scannell at slu edu.

A free spell checker for the Arabic (ar) languages called Duali can be found at `http://www.arabeyes.org/project.php?proj=duali`. It uses its own form of affix compression. This needs to be converted to Aspell format. The author, Mohammed Elzubeir, expressed an interest in converting it but I have not heard from him in a while. If you are interested in helping with this conversion please let me know.

An Ispell hash table has been discovered for Albanian (sq) at `http://www.7kosova.com/kde-shqip/ispell/ispell.html`. However, the raw word list is not provided and the author has not been responding to emails. If you know how to disassemble a Ispell hash table or the where-abouts of the word list used to create the hash table I would appreciate hearing from you.

A dictionary marked as "Planned" or "Maybe" but not listed in the section means that someone has expressed an interest in creating one. If you are interested in helping please contact me at `kevina@gnu.org` so that I can put you in touch with them.

B.5 References

The information in this chapter was gathered from numerous sources, including:

- ISO 639-2 Registration Authority, `http://www.loc.gov/standards/iso639-2/`
- Languages and Scripts (Official Unicode Site), `http://www.unicode.org/onlinedat/languages-scripts.html`
- Omniglot - a guide to written language, `http://www.omniglot.com/`
- Wikipedia - The Free Encyclopedia, `http://wikipedia.org/`
- Ethnologue - Languages of the World, `http://www.ethnologue.com/`
- World Languages - The Ultimate Language Store, `http://www.worldlanguage.com/`
- South African Languages Web, `http://www.languages.web.za/`
- The Languages and Writing Systems of Africa (Global Advisor Newsletter), `http://www.intersolinc.com/newsletters/africa.htm`

Special thanks goes to Era Eriksson for helping me with the information in this chapter.

Appendix C Language Related Issues

Here are some language related issues that a good spell checker needs to handle. If you have any more information about any of these issues, or of a new issue not discussed here, please email me at kevina@gnu.org.

C.1 Compound Words

In some languages, such as German, it is acceptable to string two words together, thus forming a compound word. However, there are rules to when this can be done. Furthermore, it is not always sufficient to simply concatenate the two words. For example, sometimes a letter is inserted between the two words. Aspell currently has support for unconditionally stringing words together. I tried implementing more sophisticated support for compound words in Aspell but it was too limiting and no one used it.

After receiving feedback from several people it seems that acceptable support for compound words involved two basically independent parts. If this is not sufficient for your language please let me know.

Part One

Describes how the word needs to be changed when forming a compound

```
CMP <flag> <strip> <add> <cond> <cond2>

<flag>  is the compound flag
<strip> is the string to strip or 0 for the null string
<add>   is the string to add or 0 for the null string
<cond>  is the condition to match at the end of the current word
<cond2> is the condition to match at the beginning of the next word
```

All but the last field are the same as a suffix entry in the exiting affix code.

<cond> is a simplified regular expression. Some examples:

```
. (for anything)
e
[^aeiou]y
[^ey]
[aeiou]y
```

It does not seem necessary to change the beginning of a word when forming compounds

Part Two

Describes the position a word can appear in (beginning, middle, or end) and with which words.

To do this each word can be assigned a category. Then each category can be given a set of rules to describe how it can be used in a compound word for example

```
A + B: indicates that category A may appear at the beginning of a
  word when followed by a category B word.  When combined it is then
  considered a category B word.
```

```
A + C + B: here a C word may only appear between an A or B word
A + A + B
A + A
A + A + A
etc..
```

I have not decided if a word should be allowed to belong to more than one category as a new category can be created in necessary to mean words in both category A and B for example.

C.1.1 To Implement

To implement support for compound words based on the above description the following will need to be done:

1. expand the affix code to support special compound flags as described in part one
2. write code to store the conditions as described in part two
3. expand the compound checking code to check against the conditions
4. expand the dictionary format to store the necessary compound info with the word

I don't know when I will be able to actually implement this. If you would like to try please let me know.

C.2 Words With Spaces or Other Symbols in Them

Many languages, including English, have words with non-letter symbols in them. For example the apostrophe. These symbols generally appear in the middle of a word, but they can also appear at the end, such as in an abbreviation. If a symbol can *only* appear as part of a word then Aspell can treat it as if it were a letter.

However, the problem is most of these symbols have other uses. For example, the apostrophe is often used as a single quote and the abbreviations marker is also used as a period. Thus, Aspell cannot blindly treat them as if they were letters.

Aspell currently handles the case where the symbol can only appear in the middle of the word fairly well. It simply assumes that if there is a letter both before and after the symbol than it is part of the word. This works most of the time but it is not fool proof. For example, suppose the user forgot to leave a space after the period:

> . . . and the dog went up the tree.Then the cat . . .

Aspell would think "tree.Then" is one word. A better solution might be to then try to check "tree" and "Then" separately. But what if one of them is not in the dictionary? Should Aspell assume "tree.Then" is one word?

The case where the symbol can appear at the beginning or end of the word is more difficult to deal with. The symbol may or may not actually be part of the word. Aspell currently handles this case by first trying to spell check the word with the symbol and if that fails, try it without. The problem is, if the word is misspelled, should Aspell assume the symbol belongs with the word or not? Currently Aspell assumes it does, which is not always the correct thing to do.

Numbers in words present a different challenge to Aspell. If Aspell treats numbers as letters than every possible number a user might write in a document must be specified in

the dictionary. This could easily be solved by having special code to assume all numbers are correctly spelled. Yet, what about something like "4th". Since the "th" suffix can appear after any number we are left with the same problem. The solution would be to have a special symbol for "any number".

Words with spaces in them, such as foreign phrases, are even more trouble to deal with. The basic problem is that when tokenizing a string there is no good way to keep phrases together. One solution is to use trial and error. If a word is not in the dictionary try grouping it with the previous or next word and see if the combined word is in the dictionary. But what if the combined word is not, should the misspelled word be grouped when looking for suggestions? One solution is to also store each part of the phrase in the dictionary, but tag it as part of a phrase and not an independent word.

To further complicate things, most applications that use spell checkers are accustom to parsing the document themselves and sending it to the spell checker a word at a time. In order to support words with spaces in them a more complicated interface will be required.

C.3 Unicode Normalization

Because Unicode contains a large number of precomposed characters there are multiple ways a character can be represented. For example letter ö can either be represented as

```
U+00F6 LATIN SMALL LETTER O WITH DIAERESIS
```

or

```
U+0061 LATIN SMALL LETTER O + U+0308 COMBINING DIAERESIS
```

By performing normalization first, Aspell will only see one of these representations. The exact form of normalization depends on the language. Give the choice of:

1. Precomposed character
2. Base letter + combining character(s)
3. Base letter only

if the precomposed character is in the target character set, then (1), if both base and combining character is present, then (2), otherwise (3).

Unicode Normalization is now implemented in Aspell 0.60.

C.4 German Sharp S

The German Sharp S or Eszett does not have an uppercase equivalent. Instead when '`ß`' is converted to '`SS`'. The conversion of '`ß`' to '`SS`' requires a special rule, and increases the length of a word, thus disallowing inplace case conversion. Furthermore, my general rule of converting all words to lowercase before looking them up in the dictionary won't work because the conversion of '`SS`' to lowercase is ambiguous; it can be '`ss`' or '`ß`'. I do plan on dealing with this eventually.

C.5 Context Sensitive Spelling

In some language, such as Luxembourgish, the spelling of a word depends on which words surround it. For example the the letter '`n`' at the end of a word will disappear if it is followed by another word starting with a certain letter such as an '`s`'. However, it can probably get more complicated than that. I would like to know how complicated before I attempt to implement support for context sensitive spelling.

Appendix D To Do

D.1 Important Items

Words in bold indicate how you should refer to the item when discussing it with me or others.

D.1.1 Things that need to be done

These items need to be done before I consider Aspell finished. If you are interested in helping me with one of these tasks please email me. Good C++ skills are needed for most of these tasks involving coding.

- Create a generic filter to handle multi-character letters such as '`"a`' or '`\"a`' for ä. This filter should make use of the already exiting normalization code if possible.
- Make Aspell **Thread safe**. Even though Aspell itself is not multi-threaded I would like it to be thread safe so that it can be used by multi-threaded programs. There are several areas of Aspell that that are potently thread unsafe (such as accessing a global pool) and several classes which have the potential of being used by more than one thread (such as the personal dictionary). *[In Progress]*.
- Enhance **ispell.el** so that it will work better with GNU Aspell. *[In Progress]*.
- Clean up copyright notices and bring the Aspell package up to **GNU Standards**. *[In Progress]*.

D.1.2 Things I would like to get done

I would like to get these done. However, I may still consider Aspell finished without. They will probably eventually get implemented. However, I could still use help with them.

- Better support for **compound words**. The support for *conditional* compound words found in Aspell versions 0.50 and earlier is no longer available since no one seems to be using it. Support for *unconditional* compound words is still available. See Section C.1 [Compound Words], page 54.
- Be able to accept **words with spaces in them** as many languages have words, such as a word in a foreign phrase, which only makes sense when followed by other words. See Section C.2 [Words With Symbols in Them], page 55.
- Reorganize manual to make it easier to understand and to make it possible to break out useful man pages.
- Support **soundslike lookup with affix compression**. I think it is possible, although I don't know how effective it will be. The basic idea is to affix compress the soundslike codes and then match the codes up with affix compressed words. If you are interested, email `aspell-devel@gnu.org`, and I will explain it in more detail.
- Use Lawrence Philips' new **Double Metaphone algorithm**. See `http://aspell.net/metaphone/`. The main task involved here is converting the algorithm into table form. This will take some time but there is no real programming experience required. If you want to help with Aspell but don't have any real programming experience, this would be a great place to start.

- Rank suggestions based on **frequency information**. Both global frequency and document specific frequency can be used. The latter will require that the whole document be made available to the spell checker. Also use frequency information to flag words which are found in the dictionary but not in common usage, and thus might not be what was intended.
- Support a **"dual-script" mode** where Aspell can use a separate dictionary depending on which script it detects the current word in, the two dictionaries can have nothing in common, ie an English one and a Russian one for example. This will *not* support two languages that use the same script as that is a lot more complicated. For example if the word is misspelled which dictionary should it use for the suggestions?
- Write a **GUI** for the Aspell utility. Ideally it should be able to do everything the Aspell utility can do and not just be able spell check a document.
- Develop a **more powerful C API** for Aspell. Ideally this API should allow one to perform all the tasks the Aspell utility can do. This included the ability to check whole documents, and create dictionaries, among other things.
- Create a **C++ interface** for Aspell, possibly on top of the C one.

D.2 Other Items

These items all sound like good ideas however I am not sure when I will get to implementing them if ever. Words in bold indicate how you should refer to the item when discussing it with me or others.

- Come up with a plug-in for `gEdit` the gnome text editor.
- Change languages (and thus dictionaries) based on the information in the actual document.
- Come up with a mode that will skip words based on the symbols that (almost) always surround the word. See Section D.3.1 [Word skipping by context], page 59.
- Create two **server modes** for Aspell. One that uses the DICT protocol and one that uses `ispell -a` method of communication via some arbitrary port.
- Come up with **thread safe personal dictionaries**.
- Use the **Hidden Markov Model** to base the suggestions on not only the word itself but on the context around the word. See Section D.3.2 [Hidden Markov Model], page 59.
- Having a way to **email the personal dictionary** and/or replacement list to a particular address either periodically or when it grows to a certain size. See Section D.3.3 [Email the Personal Dictionary], page 60.

The following good ideas were found in the ispell '`WISHES`' file so I thought I would pass them on.

- Ispell should be smart enough to ignore hyphenation signs, such as the TEX '`\-`' hyphenation indicator.
- (Jeff Edmonds) The personal dictionary should be able to remove certain words from the master dictionary, so that obscure words like "wether" wouldn't mask favorite typos.
- (Jeff Edmonds) It would be wonderful if Ispell could correct inserted spaces such as "th e" for "the" or even "can not" for "cannot".

- Since Ispell has dictionaries available to it, it is conceivable that it could automatically determine the language of a particular file by choosing the dictionary that produced the fewest spelling errors on the first few lines.

D.3 Notes on Various Items

D.3.1 Word skipping by context

This was posted on the Aspell mailing list on January 1, 1999:

I had an idea on a great general way to determine if a word should be skipped. Determine the words to skip based on the symbols that (almost) always surround the word.

For example when asked to check the following C++ code:

```
cout << "My age is: " << num << endl;
cout << "Next year I will be " << num + 1 << endl;
```

`cout`, `num`, and `endl` will all be skipped. `cout` will be skipped because it is always preceded by a '`<<`'. `num` will be skipped because it is always preceded by a '`<<`'. And `endl` will be skipped because it is always between a '`<<`' and a '`;`'.

Given the following HTML code.

```
<table width=50% cellspacing=0 cellpadding=1>
<tr><td>One<td>Two<td>Three
<tr><td>1<td>2<td>3
</table>

<table cellspacing=0 cellpadding=1>
</table>
```

`table`, `width cellspacing`, `cellpadding`, `tr`, `td` will all be skipped because they are always enclosed in '`<>`'. Now of course `table` and `width` would be marked as correct anyway however there is no harm in skipping them.

So I was wondering if anyone on this list has any experience in writing this sort of context recognition code or could give me some pointers in the right direction.

This sort of word skipping will be very powerful if done right. I imagine that it could replace specific spell checker modes for TEX, Nroff, SGML etc because it will automatically be able to figure out where it should skip words. It could also probably do a very good job on programming languages code.

If you are interested in helping me out with this or just have general comments about the idea please let me know.

D.3.2 Hidden Markov Model

Knud Haugaard Sørensen suggested this one. From his email on the Aspell mailing list:

consider these examples:

```
a fone number.
 -> a phone number.

a fone dress.
```

```
-> a fine dress.
```

the example illustrates that the right correction might depend on the context of the word.

So I suggested that you take a look on HMM to solve this problem.

This might also provide a good base to include grammar correction in Aspell.

see this link `http://www.cse.ogi.edu/CSLU/HLTsurvey/ch1node7.html`.

I think it is a great idea. However unfortunately it will probably be very complicated to implement. Perhaps in the far future.

D.3.3 Email the Personal Dictionary

Someone suggested in a personal email:

> Have you thought of adding a function to Aspell, that - when the personal dictionary has grown significantly - sends the user's personal dictionary to the maintainer of the corresponding Aspell dictionary? (if the user allows it)
>
> It would be a very useful service to the dictionary maintainers, and I think most users can see their benefit in it too.

And I replied:

> Yes I have considered something like that but not for the personal dictionaries but rather the replacement word list in order to get better test data for `http://aspell.sourceforge.net/test/`.

The problem is I don't know of a good way to do this since Aspell can also be used as a library. It also is not a real high priority, especially since I would first need to learn how to send email within a C++ program.

Appendix E Installing

Aspell requires gcc 2.95 (or better) as the C++ compiler. Other C++ compilers should work with some effort. Other C++ compilers for mostly POSIX compliant (Unix, Linux, BeOS, Cygwin) systems should work without any major problems provided that the compile can handle all of the advanced C++ features Aspell uses. C++ compilers for non-Unix systems might work but it will take some work. Aspell at very least requires a Unix-like environment ('`sh`', '`grep`', '`sed`', '`tr`', ...), and Perl in order to build. Aspell also uses a few POSIX functions when necessary.

The latest version can always be found at GNU Aspell's home page at `http://aspell.net`.

E.1 Generic Install Instructions

```
./configure && make
```

For additional `configure` options type '`./configure --help`'. You can control what C++ compiler is used by setting the environment variable `CXX` before running configure and you can control what flags are passed to the C++ compile via the environment variable `CXXFLAGS`. Static libraries are disabled by default since static libraries will not work right due to the mixing of C and C++. When a C program links with the static libraries in Aspell it is likely to crash because Aspell's C++ objects are not getting initialized correctly. However, if for some reason you want them, you can enable them via '`--enable-static`'.

Aspell should then compile without any additional user intervention. If you run into problems please first check the sections below as that might solve your problem.

To install the program simply type

```
make install
```

After Aspell is installed at least one dictionary needs to be installed. You can find them at `http://aspell.net/`. The `aspell` program must be in your path in order for the dictionaries to install correctly.

If you do not have Ispell or the traditional Unix `spell` utility installed on your system then you should also copy the compatibility scripts `ispell` and `spell` located in the '`scripts/`' directory into your binary directory which is usually '`/usr/local/bin`' so that programs that expect the `ispell` or `spell` command will work correctly.

E.2 HTML Manuals and `make clean`

The Aspell distribution includes HTML versions of the User and Developer's manual. Unfortunately, doing a `make clean` will erase them. This is due to a limitation of automake which is not easily fixed. If makeinfo is installed they can easily be rebuild with `make aspell.html aspell-dev.html`, or you can unpack them from the tarbar.

E.3 Curses Notes

If you are having problems compiling '`check_funs.cpp`' then the most likely reason is due to incompatibilities with the curses implementation on your system. You should first try disabling the "wide" curses library by with the '`--disable-wide-curses`' configure option..

By doing so you will lose support for properly displaying UTF-8 characters but you may still be able to get the full screen interface. If this fails than you can disable curses support altogether with the '`--disable-curses`' configure option. By doing this you will lose the nice full screen interface but hopefully you will be able to at least get Aspell to compile correctly.

If the curses library is installed in a non-standard location than you can specify the library and include directory with '`--enable-curses=`*`lib`*' and '`--enable-curses-include=`*`dir`*'.

'*`lib`*' can either be the complete path of the library—for example

```
/usr/local/curses/libcurses.a
```

or the name of the library (for example '`ncurses`') or a combined location and library in the form '`-L`*`libdir`* `-l`*`lib`*' (for example '`-L/usr/local/ncurses/lib -lncurses`').

dir is the location of the curses header files (for example '`/usr/local/ncurses/include`').

E.3.1 Unicode Support

In order for Aspell to correctly spell check UTF-8 documents in full screen mode the "wide" version of the curses library must be installed. This is different from the normal version of curses library, and is normally named '`libcursesw`' (with a '`w`' at the end) or '`libncursesw`'. UTF-8 documents will not display correctly without the right curses version installed.

In addition your system must also support the `mblen` function. Although this function was defined in the ISO C89 standard (ANSI X3.159-1989), not all systems have it.

E.4 Loadable Filter Notes

Support for being able to load additional filter modules at run-time has only been verified to work on Linux platforms. If you get linker errors when trying to use a filter, then it is likely that loadable filter support is not working yet on your platform. Thus, in order to get Aspell to work correctly you will need to avoid compiling the filters as individual modules by using the '`--enable-compile-in-filters`' when configuring Aspell with `./configure`.

E.5 Upgrading from Aspell 0.50

The dictionary format has changed so dictionaries will need to be recompiled.

All data, by default, is now included in '*`libdir`*`/aspell-0.60`' so that multiple versions of Aspell can more peacefully coexist. This included both the dictionaries and the language data files which were stored in '*`sharedir`*`/aspell`' before Aspell 0.60.

The format of the character data files has changed. The new character data files are installed with Aspell so you should not have to worry about it unless you made a custom one.

The dictionary option '`strip-accents`' has been removed. For this reason the old English dictionary (up to 0.51) will no longer work. A new English dictionary is now available which avoids using this option. In addition the '`ignore-accents`' option is currently unimplemented.

The flag '`-l`' is now a shortcut for '`--lang`', instead of '`--list`' as it was with Aspell 0.50.

E.5.1 Binary Compatibility

The Aspell 0.60 library is binary compatible with the Aspell 0.50 library. For this reason I chose *not* to increment the major version number (so-name) of the shared library by default which means programs that were compiled for Aspell 0.50 will also work for Aspell 0.60. However, this means that having both Aspell 0.50 and Aspell 0.60 installed at the same time can be pragmatic. If you wish to allow both Aspell 0.50 and 0.60 to be installed at the same time then you can use the configure option '`--incremented-soname`' which will increment so-name. You should only use this option if you know what you are doing. It is up to you to somehow ensure that both the Aspell 0.50 and 0.60 executables can coexist.

If after incrementing the so-name you wish to allow programs compiled for Aspell 0.50 to use Aspell 0.60 instead (thus implying that Aspell 0.50 is not installed) then you can use a special compatibility library which can be found in the '`lib5`' directory. This directory will not be entered when building or installing Aspell so you must manually build and install this library. You should build it after the rest of Aspell is built. The order in which this library is installed, with relation to the rest of Aspell, is also important. If it is installed *after* the rest of Aspell then new programs will link to the old library (which will work for Aspell 0.50 or 0.60) when built, if installed *before*, new programs will link with the new library (Aspell 0.60 only).

E.6 Upgrading from Aspell .33/Pspell .12

Aspell has undergone an extremely large number of changes since the previous Aspell/Pspell release. For one thing Pspell has been merged with Aspell so there in no longer two separate libraries you have to worry about.

Because of the massive changes between Aspell/Pspell and Aspell 0.50 you may want to clean out the old files before installing the the new Aspell. To do so do a '`make uninstall`' in the original Aspell and Pspell source directories.

The way dictionaries are handled has also changed. This includes a change in the naming conventions of both language names and dictionaries. Due to the language name change, your old personal dictionaries will not be recognized. However, you can import the old dictionaries by running the `aspell-import` script. This also means that dictionaries designed to work with older versions of Aspell are not likely to function correctly. Fortunately new dictionary packages are available for most languages. You can find them off of the Aspell home page at `http://aspell.net`.

The Pspell ABI is now part of Aspell except that the name of everything has changed due to the renaming of Pspell to Aspell. In particular please note the following name changes:

```
pspell -> aspell
manager -> speller
emulation -> enumeration
master_word_list -> main_word_list
```

Please also note that the name of the '`language-tag`' option has changed to '`lang`'. However, for backward compatibility the '`language-tag`' option will still work.

However, you should also be able to build applications that require Pspell with the new Aspell as a backward compatibility header file is provided.

Due to a change in the way dictionaries are handled, scanning for '`.pwli`' files in order to find out which dictionaries are available will no longer work. This means that programs

that relied on this technique may have problems finding dictionaries. Fortunately, GNU Aspell now provided a uniform way to list all installed dictionaries via the c API. See the file '`list-dicts.c`' in the '`examples/`' directory for an example of how to do this. Unfortunately there isn't any simple way to find out which dictionaries are installed which will work with both the old Aspell/Pspell and the new GNU Aspell.

E.7 Upgrading from a Pre-0.50 snapshot

At the last minute I decided to merge the '`speller-util`' program into the main '`aspell`' program. You may wish to remove that '`speller-util`' program to avoid confusion. This also means that dictionaries designed to work with the snapshot will no longer work with the official release.

E.8 WIN32 Notes

E.8.1 Getting the WIN32 version

The latest version of the native Aspell/WIN32 port, including binaries, can be found at `http://aspell.net/win32`. This page has, unfortunately, not been updated for Aspell 0.60. If you are interested in updated the native port please let me know.

E.8.2 Building the WIN32 version

There are two basically different ways of building Aspell using GCC for WIN32: You can either use the Cygwin compiler, which will produce binaries that depend on the POSIX layer in '`cygwin1.dll`'. The other way is using MinGW GCC, those binaries use the native C runtime from Microsoft (MSVCRT.DLL).

E.8.2.1 Building Aspell using Cygwin

This works exactly like on other POSIX compatible systems using the '`./configure && make && make install`' cycle. Some versions of Cygwin GCC will fail to link, this is caused by an incorrect '`libstdc++.la`' in the '`/lib`' directory. After removing or renaming this file, the build progress should work (GCC-2.95 and GCC-3.x should work).

E.8.2.2 Building Aspell using MinGW

There are several different ways to build Aspell using MinGW. The easiest way is to use a Cygwin compiler but instruct it to build a native binary rather than a Cygwin one. To do this configure with:

```
./configure CFLAGS='-O2 -mno-cygwin' CXXFLAGS='-O2 -mno-cygswin'
```

You may also want to add the option '`--enable-win32-relocatable`' to use more windows friendly directories. See [Win32-Directories], page 65. In this case configure with:

```
./configure CFLAGS='-O2 -mno-cygwin' CXXFLAGS='-O2 -mno-cygswin' --enable-win32-relocatable
```

It should also be possible to build Aspell using the MSYS environment. But this has not been very well tested. If building with MSYS *do not* add '`CFLAGS ...`' to configure.

E.8.2.3 Building Aspell without using Cygwin or MSYS

It is also possible to build Aspell without Cygwin of MinGW by using the files in the '`win32/`' subdirectory. However, these files have not been updated to work with Aspell

0.60. Thus the following instructions will not work without some effort. If you do get Aspell to compile this way please send me the updated files so that I can include them with the next release.

To compile Aspell with the MinGW compiler, you will need at least GCC-3.2 (as shipped with MinGW-2.0.3) and some GNU tools like `rm` and `cp`. The origin of those tools doesn't matter, it has shown to work with any tools from MinGW/MSys, Cygwin or Linux. To build Aspell, move into the '`win32`' subdirectory and type '`make`'. You can enable some additional build options by either commenting out the definitions at the head of the Makefile or passing those values as environment variables or at the `make` command line. Following options are supported:

'`DEBUGVERSION`'
: If set to `"1"`, the binaries will include debugging information (resulting in a much bigger size).

'`CURSESDIR`'
: Enter the path to the pdcurses library here, in order to get a nicer console interface (see below).

'`MSVCLIB`'
: Enter the filename of MS '`lib.exe`' here, if you want to build libraries that can be imported from MS Visual C++.

'`WIN32_RELOCATABLE`'
: If set to `"1"`, Aspell will detect the prefix from the path where the DLL resides (see below for further details).

'`TARGET`'
: Sets a prefix to be used for cross compilation (e.g. '`/usr/local/bin/i586-mingw32msvc-`' to cross compile from Linux).

There are also a MinGW compilers available for Cygwin and Linux, both versions are able to compile Aspell using the prebuilt '`Makefile`'. While the Cygwin port automatically detects the correct compiler, the Linux version depends on setting the `TARGET` variable in the '`Makefile`' (or environment) to the correct compiler prefix.

Other compilers may work. There is a patch for MS Visual C++ 6.0 available at `ftp://ftp.gnu.org/gnu/aspell`, but it needs a lot of changes to the Aspell sources. It has also been reported that the Intel C++ compiler can be used for compilation.

E.8.3 (PD)Curses

In order to get the nice full screen interface when spell checking files, a curses implementation that does not require Cygwin is required. The PDCurses (`http://pdcurses.sourceforge.net`) implementation is known to work, other implementations may work however they have not been tested. See the previous section for information on specifying the location of the curses library and include file.

Curses notes:

- PDcurses built with MinGW needs to be compiled with '`-DPDC_STATIC_BUILD`' to avoid duplicate declaration of '`DllMain`' when compiling '`aspell.exe`'.
- The curses enabled version can cause trouble in some shells (MSys `rxvt`, `emacs`) and will produce errors like '`initscr() LINES=1 COLS=1: too small`'. Use a non-curses version for those purposes.

E.8.4 Directories

If Aspell is configured with '`--enable-win32-relocatable`' or compiled with '`WIN32_RELOCATABLE=1`' when using a Makefile, it can be run from any directory: it will set '*`prefix`*' according to its install location (assuming it resides in '*`prefix`*`\\bin`'). Your personal wordlists will be saved in the '*`prefix`*' directory with their names changed from '`.aspell.`*`lang`*`.*`' to '*`lang`*`.*`' (you can override the path by setting the `HOME` environment variable).

E.8.5 Installer

The installer registers the DLLs as shared libraries, you should increase the reference counter to avoid the libraries being uninstalled if your application still depends on them (and decrease it again when uninstalling your program). The reference counters are located under:

```
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\SharedDLLs
```

The install location and version numbers are stored under

```
HKLM\SOFTWARE\Aspell
```

E.8.6 WIN32 consoles

The console uses a different encoding than GUI applications, changing this to to a Windows encoding (e.g. 1252) is not supported on Win9x/Me. On WinNT (and later) those codepages can be set by first changing the console font to '`lucida console`', then changing the codepage using '`chcp 1252`'.

Some alternative shells (e.g. MSys' `rxvt` or Cygwin's `bash`) do a codepage conversion (if correctly set up), so running Aspell inside those shells might be a workaround for Win9x.

Appendix F ChangeLog

Changes from 0.60.3 to 0.60.4 (October 19, 2005)

- Fixed a bug that caused Aspell to crash when checking certain Russian words, this bug likely affected other languages as well.
- Updated to Gettext 0.14.5 which is required for AMD64, also updated to to Libtool 1.5.20.
- Fixed an alignment bug which causes mmap to always fail when reading in dictionaries.
- Added note about how `make clean` will remove the HTML manuals.
- Added manual page for prezip-bin and enhanced word-list-compress manual page thanks to the work of Jose Da Silva.
- Other minor updates and bug fixes.

Changes from 0.60.2 to 0.60.3 (June 28, 2005)

- Fixed bugs involving several of the C API functions.
- Fixed bug where '`ultra`' or '`fast`' mode will not return any suggestions when soundslike lookup is not used.
- Made a minor, yet significant, optimization to the suggestion code. This speed things up by an order of magnitude in some cases.
- Avoid using the slow ngram scan except when the '`sug-mode`' is '`slow`' or '`bad-speller`'.
- Fixed a bug in curses mode which causes word-wrap to not work correctly in some cases.
- Fixed a bug in pipe mode with a missing newline.
- Fixed the `spell` compatibility script.
- Several other minor bugs fixed.
- Made note about the change in behavior of the '`-l`' command line switch.
- Other manual update/fixes.
- Updated to Libtool 1.5.18, Automake 1.9.6, and Makeinfo 4.8.

Changes from 0.60.1 to 0.60.2 (December 18, 2004)

- Added the `munch-list` command to the aspell utility. The `munch` program in the '`myspell/`' directory will disappear in Aspell 0.61. The `munchlist` script will also likely disappear or be replaced when Aspell 0.61 is released since it doesn't work correctly anyway.
- Several important bug fixes some of which rendered some non-English languages unusable.
- Other minor changes.

Changes from 0.60.1 to 0.60.1.1 (November 20, 2004)

- Fix bug involving checking of capitalized word when affix compression is used.
- Compile fixes.
- Added an option to disable using the "wide" curses version in case it causes compile problems.
- Minor manual updates
- Avoided including some unnecessary files in the distribution.

Changes from 0.60 to 0.60.1 (November 7, 2004)

- Lots of compile fixes for various platforms.
- Miscellaneous bug fixes.
- Added Nroff filter thanks to Sergey Poznyakoff.
- The default filter mode when in pipe mode is now nroff for compatibility with Ispell.
- Added Texinfo filter.
- Added a section detailing the differences between Ispell and Aspell.
- Updated the section on thread safety.
- Other miscellaneous manual changes such as updating the To Do and Authors section.

Changes from 0.50.5 to 0.60 (August 27, 2004)

- Added support for Affix Compression. Affix compression stores the root word and then a list of prefixes and suffixes that the word can take, and thus saves a lot of space. The codebase comes from MySpell found in OpenOffice. It uses the same affix file that OpenOffice (and Mozilla) use. Affix compression will even work with soundslike lookup to a limited extent.
- Added support for accepting all input and printing all output in UTF-8 or some other encoding different from the one Aspell uses. This includes support for Unicode normalization. Aspell can now support any language with no more than 210 distinct characters, including different capitalizations and accents, *even if* there is not an existing 8-bit encoding that supports the language.
- Added support for loadable filters and customizable filter modes thanks to Christoph Hintermüller.
- Enhanced SGML filter to also support skipping sgml tags such as "script" blocks thanks to Tom Snyder.
- Added gettext support thanks to Sergey Poznyakoff
- Reworked the compiled dictionary format. Compiled dictionaries now take up less space (less than 80% for the English language) and creating them is significantly faster (over 4 times for the English language).
- Reworked suggestion code. It is significantly faster when dealing with short words (up to 10 times). Also added support for MySpell Replacement Tables and n-gram lookup. In addition, added basic support for compound words.
- Manual has has been converted to texinfo format thanks to the work of Chris Martin.

- Reworked the build system so that a single Makefile is used for most of the code.
- All data, by default, is now included in '*libdir*`/aspell-0.60`'. Also added a built time option to increment the major version number of the shared library. This should allow both Aspell version 0.50 and 0.60 to coexist. The major version number is *not* incremented by default as Aspell 0.60 is binary compatible with Aspell 0.50. See [Binary Compatibility], page 62.
- The code to handle dictionaries has been rewritten. Because of this support for the dictionary option '`strip-accents`' has been removed. In addition the '`ignore-accents`' option is currently unimplemented.
- Lots of other minor changes due to massive overhaul of the source code.

Changes from 0.50.4.1 to 0.50.5 (Feb 10, 2004)

- Reworked url filter which fixed several bugs and now accepts "bla.bla/kdkdl" as a url.
- Fixed bug in which the url filter was coming before all other filters when it was suppose to come after. This solved a number of problems where the url filter was interfering with other filters.
- Small bug fix in SGML filter.
- Added code page charsets, ie cp125?.dat.
- Added natural (split) keyboard data file as "split.kbd"
- Compile fixes for the upcoming Gcc 3.4
- Removed Solaris link hack as it was causing more problems than it fixes.
- Compile fixes for Sun WorkShop 6 compiler, but there may still be some problems, especially with linking.
- Included patch to help compile with Microsoft Visual C++ 6.
- Minor manual fixes.
- Updated the TODO section to reflect the current progress with the next major version of Aspell (0.51).
- Updated to Autoconf 2.59, Automake 1.82, and Libtool 1.5.2.

Changes from 0.50.4 to 0.50.4.1 (Oct 11, 2003)

- Fixed major bug in pipe mode which caused the last character to be chopped off words before they were stored.
- Minor formating fixes in the manual.

Changes from 0.50.3 to 0.50.4 (Sep 26, 2003)

- Minor changes in URL filter to avoid treating the double quote character as part of the URL, and to avoid treating words ending in more than one period as a URL.
- Document fixes in Aspell API
- Small compile fixes, including one for GCC 3.3
- Updated Win32 section since a port now exists thanks to Thorsten Maerz.
- Complain instead of doing nothing or aborting for unimplemented functions in Aspell utility.

- Portability bug fixes.
- Upgraded to Autoconf 2.57, Automake 1.7.7, Libtool 1.5 (no longer use CVS version of libtool).

Changes from 0.50.2 to 0.50.3 (Nov 23, 2002)

- Hopefully fixed the ispell alignment error problem when Aspell is used with ispell.el.
- Fixed a problem with personal dictionaries on NFS mounted home directories.
- Compiled libaspell-common directory into libaspell for now to avoid forcing applications to relink whenever a new Aspell version is out which was due to the use of the libtool '-release' flag.
- Fixed Makefiles so that Aspell can be built outside the source tree (i.e. with VPATH).
- Updated the section on compiling with Win32.
- Updated to Autoconf 2.56.

Changes from 0.50.1 to 0.50.2 (Sep 28, 2002)

- Fixed a number of bugs in ispell compatibility mode
- Fixed a number of bugs with the handling of replacement pairs
- Other miscellaneous bug fixes
- Additional Win32 portability fixes
- Added the Ukrainian KOI8-U charset.

Changes from 0.50 to 0.50.1 (Aug 28, 2002)

- A rather large number of portability fixes for non GNU/Linux platforms.
- Fixed pkglibdir and pkgdatadir in configure.
- Reintroduced some configure options from Aspell .33.7 included dict-dir, data-dir, curses, curses-include, win32-relocatable.
- Fixed Aspell so it will now compile with -O3 when using gcc.
- Updated note on Win32 support.
- Other minor manual improvements.
- Portability fixes in dictionary files
- Official dictionary package for the Slovak language.

Changes from .33.7.1 to 0.50 (Aug 23, 2002)

- A complete overhaul of the source code which included merging Pspell into Aspell.
- Changed the way dictionaries and languages are handled.
- Added Dvorak keymap.
- Added the ability to list the available dictionaries
- Improved the spell checking interface a bit.
- Added support for using the Ispell keymapping when checking files.

- Complete rewrite of the filter interface. It should now be fairly easy to add new filters to Aspell.
- Added some preliminary developer documentation.
- Lots of other changes due to the massive overhaul of the source code.

Changes from .33.7 to .33.7.1 (Aug 20, 2001)

- Minor manual fixes.
- Compile fix for Gcc 3.0 and Solaris.

Changes from .33.6.3 to .33.7 (Aug 2, 2001)

- Updates to Autoconf 2.50 and switched to the HEAD branch of libtools.
- Fixed a bug which caused Aspell to crash when typo-analysis is not used such as when sug-mode is **fast** or **bad spellers**.
- Added support for typo-analysis even when a soundslike is not used.
- Fixed a bug which causes extended charters to display incorrectly on some platforms
- Compile fixes so that it will compile with Gcc 3.0.
- Compile fixed which should allow Aspell to compile with Egcs 1.1. I have not been able to actually test it though. Please let me know at kevina@users.sourceforge.net if you have have tried with Egcs 1.1.
- Compile and configuration script fixes so that USE_FILE_INO will properly be defined and Aspell will compile correctly when it is defined.
- More ANSI C++ compliance fixes.

Changes from .33.6.2 to .33.6.3 (June 3, 2001)

- Fixed a build problem in the manual/ directory by including manual-text and manual-html in the distribution.

Changes from .33.6.1 to .33.6.2 (June 3, 2001)

- Compile fix so that Aspell will work correctly when not installed in /usr/local.
- Avoided regenerating the manual unless configured with enable-maintainer-mode.
- Added the missing documentation files in the scowl directory.

Changes from .33.6 to .33.6.1 (May 29, 2001)

- Fixed a formating problem with the manual involving <.
- Added a note about creating pwli files.
- Removed the space after between the -L and the directory name in the pspell-module/Makefile which caused problems on some platforms.
- Added the configure option AM_MAINTAINER_MODE to avoid enabling rules which often causes generated build files to be rebuild with the wrong version of Libtool by default. I don't know why I didn't think to do this a long time ago.

Changes from .33.5 to .33.6 (May 18, 2001)

- Fixed a minor bug where some words would have random compound tags attached to them.
- Fixed a compile problem on many platforms where fileno is defined as a macro.
- Updated the description for a few of Aspell's options.
- Removed the note of Aspell not being able to run when compiled with the upcoming Gcc 3.0 compiler as things seam to work now.
- Added a note about Aspell not being able to compile with Egcs 1.1.
- Added hack to deal with Libtool's interdependencies problem. See bug #416981 for Pspell for more info.

Changes from .33 to .33.5 (April 5, 2001)

- **dump master** correctly detects which dictionary and language to use based on the `LANG` environment variable.
- Fixed a problem on Win32 which involves path names that begin with <Drive Letter>:.
- Bug fixes and enhancements so that Aspell can once again run under MinGW. You can even use the new full screen interface if Aspell is compiled with PDCurses.
- Some major modifications to make Aspell more C++ compliant in order to get Aspell to compile under the upcoming Gcc 3.0 compiler. This included only using STL features found in the standard version of C++. (Which means Aspell will no longer require using the SGI version of the STL) This should also make compiling C++ under non-gcc compilers a lot simpler. Please note that Aspell still has some problems with the upcoming Gcc 3.0 compiler.
- Minor changes to remove some -Wall warnings.
- Added a hack to that Aspell will properly compile as a shared library under Solaris.
- Added a few import missing words to the English word list.

Changes from .32.6 to .33 (January 28, 2001)

- Added a new new curses based interface to replace the dumb terminal interface everyone has been bitching about.
- Added the ability to give higher priority to words such as "the" instead of "teh" which are likely to be due to typos.
- Reorganized the manual so that it is hopefully easier to follow.
- Ability to automatically select the best dictionary to used based on the setting of the `LANG` environment variable.
- Expanded the medium dictionary size to include more words which included the original words found in ispell and eliminated the large size for now.
- Added three special variant add-on dictionaries.
- Switched to the multi-language branch of the CVS version of libtool.

Changes from .32.5 to .32.6 (Nov 8, 2000)

- Fixed a bug that where Aspell will crash when reading in accented characters on some platforms. This fixes bug # 112435.
- Fixed some other bugs so that it will run under Win32 under CygWin. Unfortunately it still won't run properly under Mingw.
- Fixed the mmap test in configure so that it won't fail on some platforms that use munmap(char *, int) instead of munmap(void *, int).
- Upgraded to the latest CVS version of libtool which fixed the problem with using GNU Make under Solaris.
- Added an option to copy files instead of using symbolic links for the special **multi** dictionary files.

Changes from .32.1 to .32.5 (August 18, 2000)

- Changed my email from kevinatk at home com to kevina at users sourceforge net please make a note of the new email address.
- Added an option to control if the personal replacement dictionary is saved when the save_all_wls method is called.
- Brought back the ability to dump the master word list even in the case of the special **multi** lists.
- Added a large number of hacker related words and some other slang terms to the medium size word list.
- Added an **ispell** and **spell** compatibility script for systems which don't have ispell installed. They are located in the scripts/ directory and are not installed by default.
- Manual fixes.
- Added a note on not using GNU Make on Solaris.

Changes from .32 to .32.1 (August 5, 2000)

- Minor compile fixes for recent gcc snapshot.
- Fixed naming of pwli files.
- Fixed a bug when aspell will crash when used with certain single letter flags. This bug was most noticeable when used with Emacs.
- Word list changes, see SCOWL Readme.
- Other miscellaneous changes.

Changes from .31.1 to .32 (July 23, 2000)

- Added support for optionally doing without the soundslike data.
- Greatly reduced the amount of memory used when creating word lists.
- Added support for ignoring accents when coming up with suggestions.
- Added support for local-data-dir which is searched before data-dir.
- Added support for specifying which words may be used in compounds and where they may be used.

- Added support for having more than one main word list as well as a special **multi** word list files which will allow multiple word lists to be treated as one.
- Aspell now uses a completely new word list.
- The apostrophe (') is no longer considered part of the word when it as at the end of the word such as in '`dogs'`'.

Changes from .31 to .31.1 (June 18, 2000)

- Fixed a bug where Aspell would not create a complete dictionary file on some platforms when the data is 8-bit.
- Added a workaround so Aspell will work with ispell.el 3.3.
- Minor compile fixes so it would compile better with the very latest gcc (CVS Version).
- Removed note about compiling in Win32 as I was now able to get it to work.

Changes from .30.1 to .31 (June 11, 2000)

- Added support for spell checking run together words.
- Added an option to produce a list of misspelled words from standard input.
- More robust error reporting when reading in language data files.
- Fixed a bug when that will cause Aspell to crash if the **special** line was not defined in the language data file.
- Update Pspell Module.
- Minor bug fixes.
- Added cross references in "The Aspell Utility Chapter" for easier use.

Changes from .30 to .30.1 (April 29, 2000)

- Ported Aspell to Win32 platforms.
- Portability fixes which may help aspell compile on other platforms.
- Aspell will no longer fail if for some reason the mmap fails, instead it will just read the file in as normal and free the memory when done.
- Minor changes in the format of the main word list as a result of the changes, the old format should still work in most cases.
- Fixed a bug when aspell was ignoring the extension of file names such as .html or .tex when checking files.
- Fixed a bug when aspell will go into an infinite loop when creating the main word list from a word list which has duplicates in it.
- Minor changes to the manual for better clarity.

Changes from .29.1 to .30 (April 2, 2000)

- Fixed many of the capitalization bugs found in previous versions of Aspell.
- Changed the format of the main word list yet again.
- Fixed a bug so that `aspell check` will work on the PowerPC.

- Added ability to change configuration options in the middle of a session.
- Added words from /usr/dict/words found on most Linux systems as well as a bunch of commonly used abbreviation to the word list.
- Fixed a bug when aspell will dump core after reporting certain errors when compiled with gcc 2.95 or higher. This involved reworked the Exception heritage to get around a bug in gcc 2.95.
- Added a few more commands to the list of default commands the TEX filter knows about.
- Aspell will now check if a word only contains valid characters before adding it to any dictionaries. This might mean that you have to manually delete a few words from your personal word list.
- Added option to ignore case when checking a document.
- Adjusted the parameters of the **normal** suggest mode to so that significantly less far fetched results are returned in cases such as tomatoe, which went from 100 suggestions down to 32, at the expense of getting slightly lower results (less than 1%),
- Improved the edit distance algorithm for slightly faster results.
- Removed the '`$$m`' command in pipe mode, you should now use '`$$cs mode,`*`mode`*' to set the mode and **$$cr mode** to find out the current mode.
- Reworked parts of Aspell to use Pspell services to avoid duplicating code.
- Added a module for the newly released Pspell. It will get installed with the rest of aspell.
- Miscellaneous other bug fixes.

Changes from .29 to .29.1 (Feb 18, 2000)

- Improved the TEX filter so that it will accept '`@`' at the begging of a command name and ignore trailing '*'s. It also now has better defaults for which parameters to skip.
- Reworked the main dictionary so that it can be memory mapped in. This decreases startup time and allows multiple aspell processes to use the same memory for the main word list. This also also made Aspell 64 bit clean so that it should work on an alpha now.
- Fix so that aspell can compile on platforms that gnu as is not available for.
- Fixed issue with flock so it would compile on FreeBSD.
- Minor changes in the code to make it more C++ compliant although I am sure there will still be problems when using some other compiler other than gcc or egcs.
- Added some comments to the header files to better document a few of the classes.

Changes from .28.3 to .29 (Feb 6, 2000)

- Fixed a bug in the pipe mode with lines that start with '`^$$`'.
- Added support for ignoring all words less than or equal to a specified length
- New soundslike code based thanks to the contribution of Björn Jacke. It now gets all of its data from a table making it easier for other people to add soundslike code for their native language. He also converted the metaphone algorithm to table form, eliminating the need for the old metaphone code.

- Major redesign of the suggestion code for better results.
- Changed the format of the personal word lists. In most cases it should be converted automatically.
- Changed the format of the main word list.
- Name space cleanup for more consistent naming. I now use name spaces which means that gcc 2.8.* and egcs 1.0.* will no longer cut it.
- Used file locks when reading and saving the personal dictionaries so that it truly multiprocess safe.
- Added rudimentary filter support.
- Reworked the configuration system once again. However, the changes to the end user who does not directly use my library should be minimal.
- Rewrote my code that handles parsing command line parameters so that it no longer uses popt as it was causing to many problems and didn't integrate well with my new configuration system.
- Fixed pipe mode so that it will properly ignore lines starting with '~' for better ispell compatibility.
- Aspell now has a new home page at `http://aspell.sourceforge.net/`. Please make note of the new URL.
- Miscellaneous manual fixes and clarifications.

Changes from .28.2.1 to .28.3 (Nov 20, 1999)

- Fixed a bug that caused aspell to crash when spell checking words over 60 characters long.
- Reworked **aspell check** so that
  1. You no longer have to hit enter when making a choice.
  2. It will now overwrite the original file instead of creating a new file. An optional backup can be made by using the -b option.
- Fixed a few bugs in data.cc.

Changes from .28.2 to .28.2.1 (Aug 25, 1999)

- Fixed the version number for the shared library.
- Fixed a problem with undefined references when linking to the shared library.

Changes from .28.1 to .28.2 (Aug 25, 1999)

- Fixed a bunch of bugs in the language and configuration classes.
- Minor changed in the code so that it could compile with the new gcc 2.95(.1).
- Changed the output of `dump config` so that default values are given the value `<default>`. This means that the output can be used to created a configuration file.
- Added notes on using aspell with VIM.

Changes from .28 to .28.1 (July 27, 1999)

- Removed some debug output
- Changed notes on compiling with gcc 2.8.* as I managed to get it to compile on my school account
- Avoided included **stdexcept** in '`const_string.hh`' so that I could get to compile on my schools account with gcc 2.8.1.

Changes from .27.2 to .28 (July 25, 1999)

Provided an iterator for the replacement classes.

- Added support for dumping and creating and merging the personal and replacement word lists.
- Changed the aspell utility command line a bit, it now used popt.
- Totally reworked aspell configuration system. Now aspell can get configuration from any of 5 sources: the command line, the environment variable `ASPELL_CONF`, the personal configuration file, the global configuration file, and finally the compiled in defaults.
- Totally reworked the language class in preparation for my new language code. See `http://aspell.sourceforge.net/international/` for more information of what I have in store.
- Added some options to the configure script: –enable-dict-dir=DIR, –enable-doc-dir=DIR, –enable-debug, and –enable-opt
- Removed some old header files.
- Reorganized the directory structure a bit
- Made the text version of the manual pages slightly easier to read
- Used the '`\url`' command for urls for better formating of the printed version.

Changes from .27.1 to .27.2 (Mar 1, 1999)

- Fixed a major bug that caused aspell to dump core when used without any arguments
- Fixed another major bug that caused aspell to do nothing when used in interactive mode.
- Added an option to exit in Aspell's interactive mode.
- Removed some old documentation files from the distribution.
- Minor changes on to the section on using Aspell with egcs.
- Minor changes to remove -Wall warnings.

Changes from .27 to .27.1 (Feb 24, 1999)

- Fixed a minor compile problem.
- Updated the section on using Aspell with egcs to it it more clear why the patch is necessary.

Changes from .26.2 to .27 (Feb 22, 1999)

- Totally reworked the C++ library which means you may need to change some things in your code.
- Added support for detachable and multiple personal dictionaries in the C++ class library.
- The C++ class library now throws exceptions.
- Reworked aspell ability to learn from users misspellings a bit so that it now has a memory. For more information see Section 6.3 [Notes on Storing Replacement Pairs], page 30.
- Upgraded autoconf to version 2.13 and automake to version 1.4 for better portability.
- Fixed the configuration so the `make dist` will work. For now on aspell will be distributed with `make dist`.
- Added support to skip over URL's, email addresses and host names.
- Added support for dumping the master and personal word list. You can now also merge a personal word list. Type aspell -help for help on using this feature.
- Reorganized the source code.
- Started using proper version numbers for the shared library.
- Fixed a bug that caused a spell to crash when adding certain replacement pairs.
- Fixed the problem with duplicate lines when exiting pipe mode for good.

Changed from .26.1 to .26.2 (Jan 3, 1998)

- Fixed another compile problem. Hopefully this time it will really compile OK on other peoples machines.

Changed from .26 to .26.1 (Jan 3, 1998)

- Fixed a small compile problem in '`as_data.cc`'.

Changed from .25.1 to .26 (Jan 3, 1999)

- Fixed a bug that causes duplicates items to be displayed in the suggestion list for good. (If it still does it please send be email.)
- Added the ability for aspell to learn form the users misspellings.
- Library Interface changes. Still more to come ....
- Is now multiprocess safe. When a personal dictionary (or replacement list) is saved it will now first update he list against the dictionary on disk in case another process modified it.
- Fixed the bug that caused duplicate output when used non interactively in pipe mode.
- Dropped support for gcc 2.7.2 as the C++ compiler.
- Updated the How Aspell Works (Section A.1 [Aspell Suggestion Strategy], page 42.)
- Added support for the `ASPELL_DATA_DIR` environment variable.

Changes from .25 to .25.1 (Dec 10, 1998)

- Fixed the version number so that Aspell reports the correct version number.
- Changed the note on gcc 2.7.2 compilers to make it clear that only the C++ compiler cannot be gcc 2.7.2, it is OK if the C compiler is gcc 2.7.2.
- Updated the TODO list and reorganized it a bit.
- Fixed the directory so that all the documentation will get installed in `${prefix}/doc/aspell` instead of half of it in `${prefix}/doc/aspell` and half of it in `${prefix}/doc/kspell`.

Changes from .24 to .25 (Nov 23, 1998)

- Total rework of how the main word list is stored. Start up time decreased to about 1/3 of what it was in .24 and memory usage decreased to about 2/3. (When used with the provided word list on a Linux system).

  Also the format and default locations of the main word list data files changed in the process and the data is now machine dependent. The personal word list format, however, stayed the same.
- Changed the scoring method to produce slightly better results with words like the vs. teh. And other simpler misspellings where two letters are swapped.
- Fixed the very unpredictable behavior of the '`*`', '`&`', '`@`' commands in the pipe mode.
- Added documentations for Aspell pipe mode (also known as `ispell -a` compatibility mode)
- Added a bunch of Aspell specific extensions to the pipe mode and documented them.
- Documented the `to_soundslike` and `soundslike` methods for the `aspell` class.
- Changed the scoring method to produce better results for words like *fone* vs *phone* and other words that have a spelling that doesn't directly relate to how the word sounds by using the phoneme equivalent of the word in the scoring of it.
- Added the `to_phoneme` and `have_phoneme` methods to the `SC_Language` class.
- Added the `to_phoneme` method to the `aspell` class.
- Added the framework for being able to learn from the users misspelling. Right now it just keep a log of replacements.
- Redid '`stl_rope-30.diff`'. For some reason the version of patch on my system refused it.
- Rewrite of the "*Using as a replacement for Ispell*" section and added the `run-with-aspell` utility as a replacement of the old method of mapping Ispell to Aspell.
- Fixed a bug that caused duplicate words to appear in the suggestion list.

Changes from .23 to .24 (Nov 8, 1998)

- Fixed my code so that it can once again compile with g++ 2.7.2.
- Rewrote the How It Works chapter.
- Rewrote the Requirement section and added noted on compiling with g++ 2.7.2.
- Added a To Do chapter.

- Added a Glossary and References chapter.
- Other minor documentation improvements.
- Internal code documentation improvements.

Changes from .22.1 to .23 (Oct 31, 1998)

- Minor documentation fixes.
- Changed the scoring strategy for words with 3 or less letters. This cut the number of words returned for these roughly in half.
- Expanded the word list to also include **american.0** and **american.1** from the Ispell distribution. It now includes **english.0**, **english.1**, **american.0** and **american.1** from the directory '`languages/english`' provided with Ispell 3.1.20.
- Added a link to the location of the latest Ispell.el in the documentation.
- Started a C interface and added some rough documentation for it.

Changes from .22 to .22.1 (Oct 27, 1998)

- Minor bug fixes. I was deleting arrays with delete rather than delete[]. I was suppressed that this had not created a problem.
- Added a simple test program to test for a memory leak present on some systems. (Only debian slink at the moment.) See the file memleak-test.cc for more info.

Changes from .21 to .22 (Oct 26, 1998)

- Major redesign or the scoring method. It now uses absolute distances rather than relative scores for more consistent results. See '`suggest.cc`' for more info.
- Suggest code rewritten is several places however the core process stayed the same.
- The `suggest_ultra` method temporarily does nothing. It should be working again by the next release.

Changes from .20 to .21 (Oct 13, 1998)

- Added documentation for aspell::Error
- Changed the library name from `libspell` to `libaspell`. It should never have been `libspell` in the first place. Sorry for the incompatibility.
- Added '`as_error.hh`' to the list of files copied to the include directory so that you can actually use the library outside of the source dir.
- Fixed bug that caused a segmentation fault with words where the only suggestions was inserting a space or hyphen such as in **ledgerline**.
- Added the **score** method to `aspell`.
- Changed the scoring method to deal with word when the user uses "f" in place of "ph" a lot better.

Changes from .11 to .20 (Oct 10, 1998)

- *Name change*. Everything that was Kspell is now Aspell. Sorry, the name Kspell was already used by KDE and I didn't want to cause any confusion.
- Fixed a bug that causes a segmentation fault when the `HOME` environment variable doesn't exist.

Changes from .10 to .11 (Sep 12, 1998)

- Overhaul of the SC_Language class
- Added documentation for international support
- Added documentation for the C++ library
- Other minor bug fixes.

Appendix G Authors

The folowing people or companies have contributed a non-trival amount of code to Aspell and thus own the Copyright to part of Aspell.

Jose Da Silva
: Bug fixes and enhancements to `word-list-compress`.

Sergey Poznyakoff
: Wrote the Nroff filter.

Tom Snyder
: Enhanced the SGML filter to also support skipping sgml tags such as "script" blocks.

Kevin B. Hendricks (and Contributers)
: Wrote MySpell which is a simpile spell checker library that supports affix compression. Aspell affix compression code is based on his code.

Christoph Hintermüller
: Added support for for loadable filters.

Melvin Hadasht
: Wrote a locale independent version of strtol and strtod. Wrote the original loadable filter support however his code has been completely rewritten by Christoph Hintermüller and Kevin Atkinson.

Björn Jacke
: Wrote the generic soundslike algorithm which gets all of its data from a file, thus eliminating almost all need for language specific code from Aspell.

Silicon Graphics Computer Systems, Inc.
Hewlett-Packard Company
: Parts of the SGI STL code were used in various places throughout the Aspell source.

In addition the authors of some of translated messages did not release their work into the Public Domain, and thus own the copyright to the translated text. See the files '`*.po`' in the '`po`' directory for more details.

The folowing people also contributed to the development of Aspell but do not own the Copyright to part of Aspell.

Sergey Poznyakoff
: Added gettext support.

Chris Martin
: Converted the manual to texinfo.

Lawrence Philips
: Wrote the original metaphone algorithm; however, he released his work into the Public Domain.

Michael Kuhn
: Converted the metaphone algorithm into C code and made some enhansments to the orignal algorithm. He also released his work into the Public Domain.

Geoff Kuenning (and contributers)

: The authors of Ispell. Many of the ideas used in Aspell, espacally with the affix code, were taken from Ispell. However none of the orignal Ispell code is used in Aspell.

Appendix H Copying

Copyright © 2000–2004 Kevin Atkinson.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

The library and utility program is copyright © 2000–2004 by Kevin Atkinson. You can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as (LGPL) published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

Certain parts of the library, as indicated at the top of the source file, are under a weaker license. However, all parts of the library are LGPL Compatible.

H.1 GNU Free Documentation License

Version 1.2, November 2002

Copyright © 2000,2001,2002 Free Software Foundation, Inc.
59 Temple Place, Suite 330, Boston, MA 02111-1307, USA

Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

0. PREAMBLE

   The purpose of this License is to make a manual, textbook, or other functional and useful document *free* in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or non-commercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

   This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

   We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

   This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and

is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

   You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

   You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

   If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

   If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

   If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time

you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements."

6. COLLECTIONS OF DOCUMENTS

   You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

   You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

   A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

   If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

   Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

   If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

   You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this

License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

    The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See `http://www.gnu.org/copyleft/`.

    Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

H.1.1 ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

```
Copyright (C)  year  your name.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.2
or any later version published by the Free Software Foundation;
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover
Texts.  A copy of the license is included in the section entitled ``GNU
Free Documentation License''.
```

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with...Texts." line with this:

```
with the Invariant Sections being list their titles, with
the Front-Cover Texts being list, and with the Back-Cover Texts
being list.
```

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

H.2 GNU Lesser General Public License

Version 2.1, February 1999

Copyright © 1991, 1999 Free Software Foundation, Inc.
59 Temple Place – Suite 330, Boston, MA 02111-1307, USA

Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

[This is the first released version of the Lesser GPL. It also counts as the successor of the GNU Library Public License, version 2, hence the version number 2.1.]

H.2.1 Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public Licenses are intended to guarantee your freedom to share and change free software—to make sure the software is free for all its users.

This license, the Lesser General Public License, applies to some specially designated software—typically libraries—of the Free Software Foundation and other authors who decide to use it. You can use it too, but we suggest you first think carefully about whether this license or the ordinary General Public License is the better strategy to use in any particular case, based on the explanations below.

When we speak of free software, we are referring to freedom of use, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish); that you receive source code or can get it if you want it; that you can change the software and use pieces of it in new free programs; and that you are informed that you can do these things.

To protect your rights, we need to make restrictions that forbid distributors to deny you these rights or to ask you to surrender these rights. These restrictions translate to certain responsibilities for you if you distribute copies of the library or if you modify it.

For example, if you distribute copies of the library, whether gratis or for a fee, you must give the recipients all the rights that we gave you. You must make sure that they, too, receive or can get the source code. If you link other code with the library, you must provide complete object files to the recipients, so that they can relink them with the library after making changes to the library and recompiling it. And you must show them these terms so they know their rights.

We protect your rights with a two-step method: (1) we copyright the library, and (2) we offer you this license, which gives you legal permission to copy, distribute and/or modify the library.

To protect each distributor, we want to make it very clear that there is no warranty for the free library. Also, if the library is modified by someone else and passed on, the recipients should know that what they have is not the original version, so that the original author's reputation will not be affected by problems that might be introduced by others.

Finally, software patents pose a constant threat to the existence of any free program. We wish to make sure that a company cannot effectively restrict the users of a free program by obtaining a restrictive license from a patent holder. Therefore, we insist that any patent license obtained for a version of the library must be consistent with the full freedom of use specified in this license.

Most GNU software, including some libraries, is covered by the ordinary GNU General Public License. This license, the GNU Lesser General Public License, applies to certain designated libraries, and is quite different from the ordinary General Public License. We use this license for certain libraries in order to permit linking those libraries into non-free programs.

When a program is linked with a library, whether statically or using a shared library, the combination of the two is legally speaking a combined work, a derivative of the original library. The ordinary General Public License therefore permits such linking only if the entire combination fits its criteria of freedom. The Lesser General Public License permits more lax criteria for linking other code with the library.

We call this license the *Lesser* General Public License because it does *Less* to protect the user's freedom than the ordinary General Public License. It also provides other free software developers Less of an advantage over competing non-free programs. These disadvantages are the reason we use the ordinary General Public License for many libraries. However, the Lesser license provides advantages in certain special circumstances.

For example, on rare occasions, there may be a special need to encourage the widest possible use of a certain library, so that it becomes a de-facto standard. To achieve this, non-free programs must be allowed to use the library. A more frequent case is that a free library does the same job as widely used non-free libraries. In this case, there is little to gain by limiting the free library to free software only, so we use the Lesser General Public License.

In other cases, permission to use a particular library in non-free programs enables a greater number of people to use a large body of free software. For example, permission to

use the GNU C Library in non-free programs enables many more people to use the whole GNU operating system, as well as its variant, the GNU/Linux operating system.

Although the Lesser General Public License is Less protective of the users' freedom, it does ensure that the user of a program that is linked with the Library has the freedom and the wherewithal to run that program using a modified version of the Library.

The precise terms and conditions for copying, distribution and modification follow. Pay close attention to the difference between a "work based on the library" and a "work that uses the library". The former contains code derived from the library, whereas the latter must be combined with the library in order to run.

H.2.2 TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License Agreement applies to any software library or other program which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Lesser General Public License (also called "this License"). Each licensee is addressed as "you".

   A "library" means a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and data) to form executables.

   The "Library", below, refers to any such software library or work which has been distributed under these terms. A "work based on the Library" means either the Library or any derivative work under copyright law: that is to say, a work containing the Library or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another language. (Hereinafter, translation is included without limitation in the term "modification".)

   "Source code" for a work means the preferred form of the work for making modifications to it. For a library, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the library.

   Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independent of the use of the Library in a tool for writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.

1. You may copy and distribute verbatim copies of the Library's complete source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and distribute a copy of this License along with the Library.

   You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Library or any portion of it, thus forming a work based on the Library, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a. The modified work must itself be a software library.

b. You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change.

c. You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.

d. If a facility in the modified Library refers to a function or a table of data to be supplied by an application program that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that, in the event an application does not supply such function or table, the facility still operates, and performs whatever part of its purpose remains meaningful.

(For example, a function in a library to compute square roots has a purpose that is entirely well-defined independent of the application. Therefore, Subsection 2d requires that any application-supplied function or table used by this function must be optional: if the application does not supply it, the square root function must still compute square roots.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Library, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Library, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Library.

In addition, mere aggregation of another work not based on the Library with the Library (or with a work based on the Library) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy of the Library. To do this, you must alter all the notices that refer to this License, so that they refer to the ordinary GNU General Public License, version 2, instead of to this License. (If a newer version than version 2 of the ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not make any other change in these notices.

   Once this change is made in a given copy, it is irreversible for that copy, so the ordinary GNU General Public License applies to all subsequent copies and derivative works made from that copy.

   This option is useful when you wish to copy part of the code of the Library into a program that is not a library.

4. You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you accompany it with the complete corresponding machine-readable source code,

which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange.

If distribution of object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place satisfies the requirement to distribute the source code, even though third parties are not compelled to copy the source along with the object code.

5. A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a "work that uses the Library". Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.

   However, linking a "work that uses the Library" with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than a "work that uses the library". The executable is therefore covered by this License. Section 6 states terms for distribution of such executables.

   When a "work that uses the Library" uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law.

   If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object file is unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the Library will still fall under Section 6.)

   Otherwise, if the work is a derivative of the Library, you may distribute the object code for the work under the terms of Section 6. Any executables containing that work also fall under Section 6, whether or not they are linked directly with the Library itself.

6. As an exception to the Sections above, you may also combine or link a "work that uses the Library" with the Library to produce a work containing portions of the Library, and distribute that work under terms of your choice, provided that the terms permit modification of the work for the customer's own use and reverse engineering for debugging such modifications.

   You must give prominent notice with each copy of the work that the Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work during execution displays copyright notices, you must include the copyright notice for the Library among them, as well as a reference directing the user to the copy of this License. Also, you must do one of these things:

   a. Accompany the work with the complete corresponding machine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable "work that uses the Library", as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions

files in the Library will not necessarily be able to recompile the application to use the modified definitions.)

b. Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (1) uses at run time a copy of the library already present on the user's computer system, rather than copying library functions into the executable, and (2) will operate properly with a modified version of the library, if the user installs one, as long as the modified version is interface-compatible with the version that the work was made with.

c. Accompany the work with a written offer, valid for at least three years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.

d. If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.

e. Verify that the user has already received a copy of these materials or that you have already sent this user a copy.

For an executable, the required form of the "work that uses the Library" must include any data and utility programs needed for reproducing the executable from it. However, as a special exception, the materials to be distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system. Such a contradiction means you cannot use both them and the Library together in an executable that you distribute.

7. You may place library facilities that are a work based on the Library side-by-side in a single library together with other library facilities not covered by this License, and distribute such a combined library, provided that the separate distribution of the work based on the Library and of the other library facilities is otherwise permitted, and provided that you do these two things:

   a. Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities. This must be distributed under the terms of the Sections above.

   b. Give prominent notice with the combined library of the fact that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

8. You may not copy, modify, sublicense, link with, or distribute the Library except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, link with, or distribute the Library is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

9. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Library or its derivative

works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Library (or any work based on the Library), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Library or works based on it.

10. Each time you redistribute the Library (or any work based on the Library), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify the Library subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties with this License.

11. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Library at all. For example, if a patent license would not permit royalty-free redistribution of the Library by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Library.

    If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply, and the section as a whole is intended to apply in other circumstances.

    It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

    This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

12. If the distribution and/or use of the Library is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Library under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

13. The Free Software Foundation may publish revised and/or new versions of the Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

    Each version is given a distinguishing version number. If the Library specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published

by the Free Software Foundation. If the Library does not specify a license version number, you may choose any version ever published by the Free Software Foundation.

14. If you wish to incorporate parts of the Library into other free programs whose distribution conditions are incompatible with these, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

15. BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE LIBRARY "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE LIBRARY IS WITH YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE LIBRARY AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE LIBRARY (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

H.2.3 How to Apply These Terms to Your New Libraries

If you develop a new library, and you want it to be of the greatest possible use to the public, we recommend making it free software that everyone can redistribute and change. You can do so by permitting redistribution under these terms (or, alternatively, under the terms of the ordinary General Public License).

To apply these terms, attach the following notices to the library. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
one line to give the library's name and an idea of what it does.
Copyright (C) year  name of author

This library is free software; you can redistribute it and/or modify it
under the terms of the GNU Lesser General Public License as published by
the Free Software Foundation; either version 2.1 of the License, or (at
your option) any later version.

This library is distributed in the hope that it will be useful, but
WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public
License along with this library; if not, write to the Free Software
Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307,
USA.
```

Also add information on how to contact you by electronic and paper mail.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the library, if necessary. Here is a sample; alter the names:

```
Yoyodyne, Inc., hereby disclaims all copyright interest in the library
'Frob' (a library for tweaking knobs) written by James Random Hacker.

signature of Ty Coon, 1 April 1990
Ty Coon, President of Vice
```

That's all there is to it!