Aspell .31 alpha
A More Intelligent Ispell

Kevin Atkinson

June 11, 2000

Contents

Chapter 1

Introduction

Aspell is an Open Source spell checker designed to eventually replace Ispell. Its main feature is that it does a much better job of coming up with possible suggestions than Ispell does. In fact recent tests shows that it even does better than Microsoft Word 97's spell checker in some cases. In addition it has both compile time and run time support for other non English languages. Aspell also a library however the recommend way to use aspell is through the Pspell library as the actual interface to the aspell library is constantly changing.

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.sourceforge.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 see section 3.6 for more information about the various suggestion modes.

If you have any other information you would like to add to this chart please contact me at kevinatk@home.com.

1.1.1 Comparison to Ispell

Features that only Aspell has

- Does a much better job with coming up with suggestions 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 users misspellings.
- Is an actual library that others programs can link to instead of having to use it through a pipe.
- 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.
- Support for detachable dictionaries so that more than one aspell class can use the same dictionary.
- Support for multiple personal dictionaries as well as support for special auxiliary dictionaries.
- Better support for run-together words.

Things that only Aspell will have real soon

- Ability to use multiple dictionaries by simply specifying it on the command line or in the configuration files.

Things that, currently, only Ispell have

- Lower memory footprint
- Support for affix compression
- Perhaps better support for spell checking (La)TeX files.
- Support for spell checking Nroff files.

Chapter 2

Getting Started

2.1 Requirements

Aspell requires egcs 1.1, or gcc 2.95 (or better), as the C++ compiler. Other standard complaint C++ might work however it will take some work as gcc is not fully complaint yet; at very least you will need to install Sgi's STL or STLPort. Unless you are using version 3.11 of SGI's STL you will need to apply a patch to fix a bug in the rope container. (See section 2.5.11 for what to do) That latest version of SGI's STL (`http://www.sgi.com/Technology/STL/`), version 3.11, fixes the bugs in rope and thus does not need to be fixed. All versions of egcs and any version of gcc before 2.95 will need this patch applied. Gcc 2.95 and latter uses version 3.11 (or better) and thus does not need the patch applied.

Aspell also requires the Portable Spell Checker Interface Library, otherwise known as Pspell, to be installed on your system before it will compile. Aspell requires version .11 or better. You can obtain the latest version of Pspell from `http://pspell.sourceforge.net/`

2.2 Obtaining

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

2.3 Support

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

2.4 Helping Out

The easiest thing you can do to help out it is too send me your **.aspell.≪lang≫.prepl** file located in your home directory every so often. (Email kevinatk@home.com) The file contains data on which word pairs aspell is unable to come up with the proper suggestion. If the file does not exist is simply means that you were only using aspell in a program such as emacs which does not communicate the replacement pairs back to aspell. Sending me the file would help me improve aspell accuracy as right now I don't have very much real data to work with.

Another thing you can do to help out is to try Aspell out on a different language as the Language support system is not very well tested.

2.5 Compiling & Installing

2.5.1 Generic Install Instructions

Before aspell is compiled Pspell must be installed. You can obtain the latest version of Pspell from `http://pspell.sourceforge.net/`.

Once Pspell is installed and you have read the sections below to take care of any special requirements for you system simply type

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

or

```
./configure --disable-static && make
```

to avoid making the static libraries on a system that supports shared libraries. For additional configure options type ./configure --help.

Aspell should then compile with out any additional user intervention. If you run into problems please first check the sections below as that might solve your problem. If it doesn't please send me a note at kevinatk@home.com with the compiler, system you are using and any error messages that were produced.

To install the program simply type

```
make install
```

And thats all there is too it.

2.5.2 General Problems

If aspell core dumps when spell checking files but seams OK otherwise please see section 2.5.11 as there is a good change that that will fix your problem.

If you get an unresolved symbol to "rope<char, allocator<char> >::_S_min_len" or anything with rope somewhere in the name (it might be mangled) try changing the line

```
//template crope;
```

to

```
template crope;
```

in aspell.cc located in the src/ directory.

2.5.3 Upgrading from version .30

The format of the main dictionary file has changed a bit. If you were able to use Aspell .30 then the old format should work. The only time the old format will NOT work is in the rare case the mmap fails. Previous version of aspell will just abort with an error when the mmap fails but the new version will attempt to read in the file using fread. Fread will fail with the old version of the main word list.

2.5.4 Upgrading from version .29.1

The format (but not the name) of the main dictionary has changed yet again. The install process will over write the old version, so unless you are using dictionaries other than the one provided with aspell or want to have multiple versions of aspell installed you should not have to worry about this.

Aspell also now depends on Portable Spell Checker Interface Library otherwise known as Pspell. Pspell must be installed before aspell will compile, you can find it at `http://pspell.sourceforge.net/`.

2.5.5 Upgrading from version .29

The format and name of the main dictionary has changed which means it will need to be recompiled. The install process will remove the old files for you, so unless you are using dictionaries other than the one provided with aspell or want to have multiple versions of aspell installed you should not have to worry about this.

2.5.6 Upgrading from version .28.3

Aspell now uses namespaces which means egcs 1.0 and gcc 2.8 will no longer cut it. If this becomes a serious problem let me know as it should not be to difficult to get it working again with egcs 1.0 and gcc 2.8.

Due to the new soundslike code the the main dictionary will need to be recompiled. The build process does this automatically so unless you want to have more than one version of aspell around you should not need to worry about this.

The format and file name of the personal dictionaries has also changed. In most cases aspell will automatically detect this and convert it for you by using the following algorithm.

1. If no file exist of *.pws (for the personal word list) or *.prepl (for the personal replacement list) aspell will look for *.per or *.rpl respectfully. If that file is found it will read in the data using the old format.
2. When saving the dictionary it will save is as *.pws or *.prepl respectfully.
3. Once saved as the new format it will delete the old file.

If you have an older version of aspell around you can restore the old dictionaries by using these command

aspell.new dump personal | aspell.old create personal

aspell.new dump repl | aspell.old create repl

The new version of aspell will then leave the old files alone as long as *.pws and *.prepl exist.

Also, if the file does not end in .pws or .prepl it will try to read it in using the new format and if that fails it will read in the old format. When the file is saved it will be saved as the new format.

I am hopping I will not have to change the format of the personal dictionaries again. However, the main word list however is very likely to change in format.

2.5.7 Upgrading from version .28.2.1

The behavior of “aspell check” changed so that it will now over right the original file as creating new file was creating too many problems when used with programs like pine and vi.

2.5.8 Upgrading from version .27.2

The name of the personal word lists have changes from .aspell.per and .aspell.rpl to .aspell.≪lang≫.per and .aspell.≪lang≫.rpl respectively. ≪lang≫ is is the language name which will generally be “english”. If you wish to use your old word lists you will need to rename those files.

2.5.9 Upgrading from version .25

The format of the personal replacement dictionary has changed. So, you will either need to rename or remove the file **.aspell.rpl** located in your home directory. If you have information in this file you would like to preserve please send me an email.

2.5.10 Upgrading from version .24

Because the location of the main word list moved you should probably do a **make uninstall** (with the old version of a Aspell) before upgrading to remove the old word lists. A make uninstall will not remove any personal word lists.

2.5.11 Compiling with Egcs and Gcc 2.8

In order for aspell to function properly under egcs or gcc 2.8.* you will need to apply the patch file stl_rope-30.diff. (Gcc 2.95 uses version 3.11 of SGI's STL and thus does not need this patch applied.) To apply the patch cd over to the g++ include directory (normally a subdirectory in the normal include directory) and type in the command

patch < ≪path≫/misc/stl_rope-30.diff

(where path is the full patch to the location where aspell was unpacked) to patch the necessary file. If it is impossible for you to do this copy the file stl_rope.h to the the directory *src/* in the aspell distribution and then type the command:

patch < ../misc/stl_rope-30.diff

Then proceed as normal.

If you do not apply this command aspell will crash when used interactively to spell check files. "aspell -a" and all other aspell modes will work fine however.

2.5.12 Win32 Notes

I was not able to get this version of Aspell to compile on Win32. Libtool failed when trying to create the libaspell library. For now the best bet is to use Aspell .30.1 and Pspell .10. If you can get it to compile on Win32 please send me a note with what you had to do.

Aspell is now able to compile on Win32 platforms using the Win32 version of gcc. Aspell can either be compiled with Cgiwin or Mingw version of gcc 2.95 using Cgiwin development environment. Versions compiled under Mingw will have slightly less functionality than the Unix version however it is very unlikely that you will notice any difference. Version compiled under Mingw don't use file locks which means there is a slight change two different aspell programs opened at the same time may try to write the personal word list at the same time leading to unpredictable results. Mingw compiled binaries also won't use mmap when loading the main word list.

The default paths for Aspell are designed for a Unix system and not a Win32 system so you might want to specify different ones when compiling Aspell. Also if the HOME environmental variable is not set aspell will assume it is the current working directly. This may lead to your personal word lists being saved in unpredictable locations. To solve this specify the complete path of the personal and replacement word lists in aspell.conf. For example the Win32 binary version of Aspell .30 had the following aspell.conf:

```
personal /aspell/dict/english.pws
repl     /aspell/dict/english.prepl
```

2.6 Using as a replacement for Ispell

Aspell can currently function 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 use simple call the ispell command and expect the original file to be overwritten with the corrected version. It support the basic features of Ispell however it does not currently have a Nroff mode so there may be situations in which you still wish to use Ispell. Nevertheless, I have been using Aspell for Xemacs and LyX since the middle of September of 1998 with out any problems.

The recommended way to use Aspell as a replacement for ispell is to change the Ispell command from within the program being uses. 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 for programs using ispell in pipe mode. 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.

2.6.1 Notes on the Learning from Users Mistakes feature

Because Ispell does not have this feature most all programs that use ispell through a pipe will not be able to take advantage of this because they do not communicate back the replacement pairs.

If you want to add support for this feature in your program the best way to do it is to scan the version string for the string "Aspell" and if found use the "$$ra" command to communicate back to Aspell the replacement pairs. See section 3.2.1 for more information.

2.6.2 Emacs and Xemacs Notes

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.

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

Unfortunately the very latest version of ispell.el, version 3.3, will not work with aspell. Hopefully, version 3.4 will work with aspell but until them our only choice is to downgrade to an earlier version.

2.6.3 LyX Notes

Version 1.0 of LyX provides support for Aspell learning for users mistake 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.

2.6.4 VIM Notes

(The following section was written by "R. Marc" <rmarc@copacetic.net>.)

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>
```

2.6.5 Pine Notes

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 peoples comments change **speller** to

    aspell check

instead which will avoid switching aspell into email mode.

Chapter 3

The Aspell utility

The Aspell utility is a multipurpose utility that can function as a "ispell -a" replacement, as an independent spell checker, and as a utility for managing dictionaries. Here is a brief summary of Aspell's command line options. For a more detailed explanation of how to use Aspell to perform a particular task please see one of the proceeding sections.

> aspell [options] ≪command≫

≪command≫ is one of:

check ≪file≫ to check a file (see section 3.1)

pipe 'ispell -a' compatible mode (see section 3.2).

list produce a list of misspelled words from standard input

config dump the current configuration to stdout

soundslike returns the soundslike equivalent for each word entered

filter passes standard input through the same set of filters that would be used to spell check a document.

help display online help

version prints a version line

dump|create|merge maser|personal|repl [word list] dumps, creates, or merges a master, personal, or replacement word list (see section 3.3).

[options] is any or all of the following standard aspell library options:

--conf=≪file≫ main configuration file

--conf-dir=≪dir≫ location of main configuration file

--data-dir=≪dir≫ location of language data files

--dict-dir=≪dir≫ location of the main word list

--add|rem-filter=≪str≫ add or removes a filter

--home-dir=≪dir≫ location for personal files

-W,--ignore=≪integer≫ ignore words <= n chars

--[dont-]ignore-case ignore case when checking words

--[dont-]ignore-repl ignore commands to store replacement pairs

--lang=≪str≫ default language to use

-d,--master=≪name≫ main word list base name

--mode=≪str≫ sets the filter mode. Mode is one if none, url, email, sgml, or tex.

-e,--mode=email enter Email mode.

-H,--mode=sgml enter Html/Sgml mode.

-t,--mode=tex enter TEX mode.

--per-conf=≪file≫ personal configuration file

-p,--personal=≪file≫ personal word list file name

--repl=≪file≫ replacements list file name

--sug-mode=≪mode≫ suggestion mode = fast | normal | bad-spellers (see section 3.6)

in addition to options to control the behavior of run-together words(see section 3.4):

-C|--run-together consider run-together words legal

-B|--dont-run-together don't consider run-together words legal

--run-together-limit=<int> maximum numbers that can be strung together

--run-together-middle=<str> letter than can appear between words

--run-together-min=<int> minimal length of interior words

plus options to modify the behavior of the various filters (see section 3.5):

--add|rem-email-quote=≪char≫ email quote characters

--email-margin=≪integer≫ num chars that can appear before the quote char

--add|rem-sgml-check=≪str≫ sgml tags to always check.

--add|rem-sgml-extension=≪str≫ sgml file extensions.

--add|rem-tex-command=≪str≫ TEX commands

--[dont-]tex-check-comments check TEX comments

in addition to some aspell utility specific command:

-b,--backup create a backup file by appending ".bak" to the file name. (Only applies when the command is **check**)

-x,--dont-backup don't create a backup file.

--[dont-]time time load time and suggest time in pipe mode.

--[dont-]reverse reverse the order of the suggestions list.

In addition Aspell with try to make seance out of Ispell's command line options so that it can function as a drop in replacement for Ispell when used is "-a" mode.

If Aspell is specified with out any command line options it will display a brief help screen and quit.

Aspell can also make use of a global or user configuration file. Each line of the configuration file has the format:

```
≪option≫ [args]
```

where option is any one of the standard library options above without the leading dashes. For example the following line will set the default language to German:

```
lang german
```

Anything from a "#" to a newline is ignored. The global configuration file is usually named "aspell.conf" and is found in the etc directory while the user configuration file is usually named ".aspell.conf" and is found in the users home directory. Use "aspell dump config" to found out what they are for your installation.

The environmental 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.

3.1 As an independent spell checker

To use Aspell as an independent spell checker type

> aspell check ≪filename≫

Where ≪filename≫ is the file you want to check. Aspell will over right the original file with the corrected version. The original version is saved as ≪filename≫.bak unless it is turned off with the dont-backup option.

If the extension is .tex in will check the file in tex mode unless overridden by the **mode** option. If the extension is one of the extensions in the **sgml-extension** option (see section 3.5.4) it will check the file in sgml unless overridden by the mode option.

The exit command saves the file with the corrections made so far. If you want to quite without saving use control-C.

3.2 As a "ispell -a" replacement

To actually use Aspell as an Ispell replacement simply follow the directions in section 2.6.

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.

3.2.1 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 ispell 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 ispell to insert the word into the user’s dictionary. A line starting with ’&’ tells ispell to insert an all-lowercase version of the word into the user’s dictionary. A line starting with ’@’ causes ispell 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 ispell in TeX/LaTeX mode (similar to the -t option) and ’-’ returns aspell to its default mode (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 ispell 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 extension. 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 command.

cs ≪option≫,≪value≫ Change a configuration option.

cr ≪option≫ Prints the value of a configuration option.

s ≪word1≫,≪word2≫ Returns the score of the two words based roughly on how aspell would score them.

Sw ≪word≫ Returns the soundlike equivalent of the word.

Sl ≪word≫ Returns a list of words that have the same soundlike equivalent.

Pw ≪word≫ Returns the phoneme equivalent of the word.

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 latter use. Returns nothing.

Anything returned is returned on its own line 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)

Notes of Storing Replacement Pairs

As of version .27 of Aspell storing replacements pairs 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.

3.3 As an utility to manage word lists

3.3.1 Creating a Word List

To create the 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 ≪word list≫ is the list of words separated by white space. 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. See chapter 5 for more information on using Aspell with other languages.

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≫.

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

Because Aspell does not support any sort of affix compression like Ispell does Ispell word lists will not work as is. In order to use Ispell's word lists simply pipe the word list through "ispell -e" to expand the munched word lists.

3.3.2 Format of the Replacement Word List

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

```
≪misspelled word≫: ≪correction≫
```

3.3.3 Dumping the contents of the word list

The dump command will simply dump the contents of a word list to stdout in a formate than 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 master

will simply dump the contents of the current master word list to stdout.

3.4 Notes on the option to control run-together words

The **run-together** options control if Aspell excepts run together words as legal compounds. The default for this option is normally false but can be changed by the particular language in used. The other **run-together-*** options control the behavior of how aspell behaves with run-together words if the run-together options in enabled, the exact defaults for these options are language dependent. The **run-together-limit** options controls the maximum number of words that can be strung together, the default is normally 255. The run-together-min options controls the minimal length the individual components of the run together word can be, the default is normally 3. The option **run-together-middle** specifies an optional character that can appear between run-together words, the default is normally none.

Currently Aspell only supports run-together words when checking if a word is in the dictionary. When coming up with suggestions Aspell treats the word as a normal word and does not do anything special. This means that the suggestions will be virtually meaningless when the actual word is a run-together. I plan on more intelligently supporting run-together words when coming up with suggestions in a future version of Aspell.

3.5 Notes on various filters and filter mode

Aspell now has rudimentary filter support. You can either select from individual filters or chose a filter mode. To select a filter mode use the **mode** option. You may chose from **none**, **url**, **email**, **sgml**, and **tex**. The default mode is **url**. Individual filters can be added with the option **add-filter** and remove with the **rem-filter** option. The currently available filters are **url**, **email**, **sgml**, **tex** as well as a bunch of filters which translate the text from one format to another.

3.5.1 None Mode

This mode is exactly what it says. It turns off all filters.

3.5.2 Url Filter/Mode

The **url** filter/mode skips over URL's, 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.

3.5.3 Email Filter/Mode

The **email** filter/mode skips over quoted text. It currently does not support skipping over headers however a future version should. In the mean time 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 char, the default is 10. The option **add|rem-email-quote** controls the characters that are considered quote characters, the default is “>’ and ’|’.

3.5.4 SGML Filter/Mode

The **sgml** filter/mode will skip over sgml commands. It currently does not handle nested < > unless they are in quotes. It also does it handle the null end tag (net) minimization feature of sgml such as

<emphasis/important/

The option **add|rem-sgml-check** controls which sgml tags should always be checked. The default is “alt”.

The option **add|rem-sgml-extension** controls which file extensions are recognized as sgml/html files. The default is html, htm, php, and sgml. The extension are not case sensitive so extensions like .HTM will also be recognized.

The sgml mode also enables a filter which will recognize sgml charter commands such as & and convert it into the proper iso8859-1 character. Currently only the iso8859-1 character set is used however in future versions it will convert it to the encoding that is specified in the language date file. You can specifically turn on this filter by enable the **SGML&≪charset≫/≪charset≫** filter.

3.5.5 TeX Filter/Mode

The **tex** (all lowercase) filter/mode skips over TeX commands and parameters and/or options to certain command. It also skips over TeX comments by default. The option **[dont-]tex-check-comments** controls whether or not aspel 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 there parameters and/or options checked. The format for each item is

≪command≫ ≪a list of p,P,o and Os≫

The first item is simple the command name. The second item controls which parameters to skip over. A ’p’ skips over a parameter while a ’P’ won’t. Similar an ’o’ will skip over an optional parameter while a ’O’ won’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.

3.6 Notes on the different suggestion modes

In order to understand what these suggestion modes do, a basic understanding of how aspell works is required. See section 6 for that. The suggestion modes are as follows.

fast This method looks for soundslikes within one edit distance apart. It is slower than Ispell by a factor of 1.5 to 2. It speed is only minor affected by the size of the word list, if at all. In this mode Aspell gets 88% of the words from my small test kernel of misspelled words. (Go to `http://aspell.sourceforge.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.)

normal This method looks for soundslikes within two edit distance apart. Is is around 10 times slower than fast mode with the english word list but returns better suggestions. Its speed is directly proportional to the size of the word list. This mode gets 93% of the words.

bad-spellers This method also looks for soundslikes within two edit distances apart but is more tailored for the bad speller where as fast or normal are more tailed to strike a good balance between typos and true misspellings. This method also returns a *huge* number of words for the really bad spellers who can’t seam 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.

Chapter 4

Library Interface

4.1 The C++ Library

The recommend way to use Aspell is through the Pspell library. Please see the Pspell manual for more details.

4.2 The C Library Interface

Aspell now has a C library interface through use of Pspell. Please see the Pspell manual for more details.

4.3 Notes About Thread Safety

Read-only Aspell methods and functions should be thread safe as long as exceptions, new, delete, delete[], and STL allocators are thread safe. To the best of my knowledge gcc and egcs meet these requirements. It is up to the programmer to make sure multiple threads do not do thing such as change the dictionaries and add or delete items from the personal or session dictionaries.

Chapter 5

International Support

Adding a language to aspell is fairly straightforward. You need to at very least create the language data file, and compile a new word list.

Please note, however, that Aspell international support is not 100% done yet. More information on my future planes for international support in aspell can be found at `http://aspell.sourceforge.net/international/`.

5.1 The Language Data File

The basic format of the language data data is the same as it for 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.

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 charset aspell will expect the word lists to be formated in. You may chose from any of the iso-8859-* character sets as well as, koi8-f, koi8-r, and viscii. If your language can fit in the plain old ASCII character set use iso8859-1. If you use some other character set for your language other than the ones listed here drop me a note and I will look into adding support for it.

The optional fields are **special, soundslike**, and a bunch of options to specify how run-together words are handles. **Special** is for 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 at the beginning of the word, a '-' means it cant. The same is true for ≪middle≫ and ≪end≫. For example the entry for the **'** in english is.

```
' -**
```

The **soundslike** option, if present, should be the name of the soundslike data for the language. The data is expected to be in the file ≪name≫_phonetic.dat. If the name is **generic** a really generic soundslike algorithm will be used which consists of striping all the vowels and removing all accents. I recommend first using the generic algorithm and then, after aspell is working with the new language, work on the transformation array.

The options to control how run-together words are handled are the same as the are in the normal configurations files. Please see section 3.4 for more information on these options.

5.2 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. See section 3.3 to find out what to do.

5.3 Phonetic Code

(The following section was written by Björn Jacke <bjoern.jacke@gmx.de>)

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 stones — 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.

5.3.1 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 when ever the 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 abort with an error if they don't. Thus it is important to update it when ever a new version of the translation array is releases. 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 (ie, 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 → _" would stand 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 transform 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.

Behind the search string there can stand one or more dashes (-). 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 more clear:

```
...
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" the 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 torn 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 self 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 first matched string in at 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 "$" only apply on words that end with the search string. "GN$ → 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.

5.3.2 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.

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 trough `\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.

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 this cases mostly one character differs from the original word.

In cases where the writer didn’t know about the correct spelling of the word however the word may have several characters that differ from the original word but usually the word would still sound like the original word. Someone might think for example 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 this 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 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 so big.

Don’t forget about follow-up rules which can help in many cases but which also can lead to many confusions and 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 control if changes you apply to your rules don’t have any evil 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 one time before the change and one time 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 for 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.

Chapter 6

How Aspell Works

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 words 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 check if each one is in the dictionary. When set to find all soundslike within two edit distance 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**.

Appendix A

Changelog

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 environmental 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. See section 3.2.1 for more information.
- 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 (6) Chapter.
- Added support for the ASPELL_DATA_DIR environmental 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 can not 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.
- Re did **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 temporally 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 environmental 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 B

To Do

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

B.1 Things that will be done real soon

These items should get done within a release or two.

- Totally rewrite the aspell international support. See `http://aspell.sourceforge.net/international/` for more information.
- Rework the aspell **check** function to provide support for using any number of filters which will be needed for international support.
- Add support for more intelligently coming up with suggestions for words that are run-togethers.

B.2 Things that need to be done

Things items will eventually be implemented as I know they are important however I am not sure when they will get done.

- Figure out a way for Aspell to work better with **extremely large dictionaries**.

B.3 Things that I would like to get done

These items will eventually be implemented. I hope to have them all done before I move aspell to beta testing. They are in the approximate order of when they will get done.

B.4 Things that will be done eventually

I plan on doing these things eventually. It is just a matter of getting around to it.

B.5 Good ideas that are worth implementing

These items all sound like good ideas however I am not sure when I will get to implementing then if ever. If you are looking for a way to contribute picking up on one of these ideas would be a great way to start. They are presented in no particular order.

- Use Lawrence Philips' new Double Metaphone algorithm. See `http://aspell.sourceforge.net/metaphone/`.
- Add support for **affix compression.**
- Come up with a **full screen terminal** interface similar to Ispell's.
- 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 **nroff mode** for spell checking.
- Come up with a mode that will skip words based on the symbols that (almost) always surround the word. (**Word skipping by context**)
- Create two **server modes** for Aspell. One that uses the **DICT** protocol and one that uses **ispell -a** method of communication of 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.

The following good ideas where 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.

B.6 Things that are not likely to get implemented

Theses ideas are not likely to get implemented any time soon.

- (None Yet)

B.7 Notes and Status of various items

B.7.1 Affix Compression

Due to the current way my spell checker works implementing affix compression would be next to impossible. Nevertheless, I do realize that for some languages affix compression is very important.

So to solve this dilemma I plan on having two different modes of my spell checker: One with affix compression that does not use soundslike pairs (much like ispell) and one without affix compression that does use soundslike.

I plan to extract the affix manipulation code from Ispell with the help of an Ispell author. The tricky part would be getting this to getting this all to work properly at tun time bases on the dictionary used.

B.7.2 Extremely Large Dictionaries

This problem extends back to the fact of the way words are index is Aspell. This problem will get resolved when I implanted the affix compression mode as only one index would be used.

B.7.3 General region skipping

I want to implement this give other people an idea of how it should be done and because I am really sick of having to spell check through url and email address.

B.7.4 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 inclosed 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 be out with this or just have general comments about the idea please let me know.

B.7.5 Hidden Markov Model

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

> consider this 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 suggest 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...

Appendix C

Support for Gcc 2.7.2

(and other non-standard compliance compilers)

My original plan was to program in such a way that it would Aspell would compile under Gcc 2.7.2 however after releasing a rather nasty bug in 2.7.2 with nested types I have desired to drop all support for Gcc 2.7.2. As of Aspell .27 all hope for being able to compile under Gcc 2.7.2 is lost as I am now using many modern C++ features which are simply not present in Gcc 2.7.2, most notably template specialization and template members. Egcs 1.1.1 is a very good standards compliant compiler and that is now the officially supported compiler. However as Egcs 1.1.1 is relatively new and, except for namespaces, provides little new functionality I will continue to support Egcs 1.0.3. Gcc 2.8.1 should in theory work however it is so buggy I have yet to get it to compile with it nor has anyone else that I know of.

Yes, my code could be rewritten so that is could compile under Gcc 2.7.2 but Why? Using modern C++ has probably accelerated the development of this library my at least 50%. And for that matter why stop at Gcc 2.7.2 why not go all out and totally rewrite my code pure C. I hope you see my point.

However that does not mean I want to sacrifice portability unnecessarily. If you see any of part of my code that in not Standard C++ please let me know. My hope is that my code could compile on all Standard compliance C++ compilers with the addition of a few extra header files from SGI's STL.

As a side note I think that Mozilla's C++ portability guide (`http://www.mozilla.org/docs/tplist/catBuild/portable-cpp.html`) could be summed up in one sentence: Program in the dark ages of C++.

Appendix D

Credits

- To the many authors of Ispell (including R. E. Gorin, Pace Willisson, and Geoff Kuenning) for providing me with a good word list as well as giving me a few good ideas.
- Lawrence Philips for coming up with the original metaphone algorithm and Michael Kuhn for writing C code for the algorithm.
- Björn Jacke for coming up with a generic soundslike algorithm which gets all of its data from a file, thus eliminating almost all need for language specific code from aspell.
- To the authors of SGI STL version 3.0 and up for proving a great set of generic container classes with cut the development time of this program in half at least.
- To the LyX development team for giving me a great tool for the development of this manual.

Appendix E

Glossary and References

affix in grammar, a word element that, when added to a word, modifies its meaning or function; prefix, infix, or suffix.

Debian A 100% Open Source Linux distribution `http://www.debian.org`

DICT protocol A TCP transaction based query/response protocol that allows a client to access dictionary definitions from a set of natural language dictionary databases. `http://www.dict.org/`

Gnome A project to build a complete, user-friendly desktop based entirely on free software. `http://www.gnome.org/`

GTK+ A library for creating graphical user interfaces for the X Window System. It is designed to be small, efficient, and flexible. `http://www.gtk.org/`

GUI Graphics User Interface

Ispell.el Emacs interface for ispell. `http://www.kdstevens.com/~stevens/ispell-page.html`

Ispell An international spell checker which is just about the only decent Open Source spell checker out there. (except Aspell of course). `http://fmg-www.cs.ucla.edu/geoff/ispell.html`

Ispell -a An Ispell mode that is designed to be used by other applications though a pipe.

KDE A powerful graphical desktop environment for Unix workstations. `http://www.kde.org`

Linux A Open Source version of Unix which runs on many platforms. `http://www.linux.org/`

LyX An What You See is What You Mean document editor for the X environment, `http://www.lyx.org`

Open Source Software where the source code is available for anyone to extend or modify. `http://www.opensource.org/`

Red Hat A commercial Linux distribution. `http://www.redhat.com`

RPM Red Hat's packing format also used by other Linux distributions.

STL Standard Temple Library. A C++ library of container classes, algorithms, and iterators.

SGI STL Silicon Graphics implantation of the STL. `http://www.sgi.com/Technology/STL/`

STLPort A port of SGI STL designed to run on compilers that don't support all the latest features of C++. `http://corp.metabyte.com/~fbp/stl/`

Appendix F

Copyright

The library and utility program is Copyrighted 2000 by Kevin Atkinson under the LGPL. Certain parts of the library, as indicated at the top of the source file, are under a weaker license. The provided word list is extracted from the word lists provided with Ispell thus it is distributed under the same terms of Ispell.

The two licenses follow:

F.1 LGPL

GNU LIBRARY GENERAL PUBLIC LICENSE
Version 2, June 1991

Copyright (C) 1991 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 library GPL. It is numbered 2 because it goes with version 2 of the ordinary GPL.]

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 Library General Public License, applies to some specially designated Free Software Foundation software, and to any other libraries whose authors decide to use it. You can use it for your libraries, too.

When we speak of free software, we are referring to freedom, 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 or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the 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 a program 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.

Our method of protecting your rights has two steps: (1) copyright the library, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the library.

Also, for each distributor's protection, we want to make certain that everyone understands that there is no warranty for this free library. If the library is modified by someone else and passed on, we want its recipients to know that what they have is not the original version, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that companies distributing free software will individually obtain patent licenses, thus in effect transforming the program into proprietary software. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

Most GNU software, including some libraries, is covered by the ordinary GNU General Public License, which was designed for utility programs. This license, the GNU Library General Public License, applies to certain designated libraries. This license is quite different from the ordinary one; be sure to read it in full, and don't assume that anything in it is the same as in the ordinary license.

The reason we have a separate public license for some libraries is that they blur the distinction we usually make between modifying or adding to a program and simply using it. Linking a program with a library, without changing the library, is in some sense simply using the library, and is analogous to running a utility program or application program. However, in a textual and legal sense, the linked executable is a combined work, a derivative of the original library, and the ordinary General Public License treats it as such.

Because of this blurred distinction, using the ordinary General Public License for libraries did not effectively promote software sharing, because most developers did not use the libraries. We concluded that weaker conditions might promote sharing better.

However, unrestricted linking of non-free programs would deprive the users of those programs of all benefit from the free status of the libraries themselves. This Library General Public License is intended to permit developers of non-free programs to use free libraries, while preserving your freedom as a user of such programs to change the free libraries that are incorporated in them. (We have not seen how to achieve this as regards changes in header files, but we have achieved it as regards changes in the actual functions of the Library.) The hope is that this will lead to faster development of free libraries.

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, while the latter only works together with the library.

Note that it is possible for a library to be covered by the ordinary General Public License rather than by this special one.

GNU LIBRARY GENERAL PUBLIC LICENSE
TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License Agreement applies to any software library which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Library 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 compile 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) 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.
> c) 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.
> d) 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 source code 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 to 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 Library 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

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 a brief 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 Library General Public License as published by the Free Software Foundation; either version 2 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 Library General Public License for more details.

> You should have received a copy of the GNU Library General Public License along with this library; if not, write to the Free 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!

F.2 Ispell Copyright