AlphaSpell

Fergus Duniho

COLLABORATORS			
	TITLE : AlphaSpell		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Fergus Duniho	July 20, 2024	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

AlphaSpell

1.1 AlphaSpell V

```
    ___    __      __            _____            ____   _    __
   /   |  / /___  / /_  ____ _  / ___/____  ___  / / /  | |  / /
  / /| | / / __ \/ __ \/ __ `/  \__ \/ __ \/ _ \/ / /   | | / /
 / ___ |/ / /_/ / / / / /_/ /  ___/ / /_/ /  __/ / /    | |/ /
/_/  |_/_/ .___/_/ /_/\__,_/  /____/ .___/\___/_/_/     |___/
        /_/                       /_/
```

1.2 Introduction

AlphaSpell V is designed to provide fast spell checking to a text editor or word processor. What is required is an editor (or word processor) that includes a macro language or has ARexx support. AlphaSpell was designed specifically for use with XDME, which has both, but it ought to be usable with such editors as Emacs, FrexxEd, GoldEd, and others. All any of these need is an interface that calls AlphaSpell and feeds its output to a requester that lets the user see his mistakes and correct them in the document. Although AlphaSpell can also be used as a stand-alone spelling checker, it isn't designed to help you correct your text. It leaves that task to the text editor itself, which can do it better. What AlphaSpell does is quickly provide the text editor with the data it needs to perform this task. My XES package provides an AlphaSpell interface for XDME, and the GUI file for it is included with AlphaSpell, so that people can use it as a model for designing other interfaces. So that I may spread the use of AlphaSpell, the first people who provide me with solid interfaces for other text editors will become registered users of AlphaSpell without having to pay the registration fee.

AlphaSpell’s name comes from the manner in which is spell checks a document. It makes an alphabetized list of all the words in a document and passes them through a dictionary in alphabetical order. Actually, it makes two lists, one for case sensitive spell checking and the other for case insensitive spell checking. AlphaSpell uses mixed case dictionaries for words whose case matters, such as names, acronyms, and German nouns, and it uses lowercase dictionaries for words whose case doesn’t matter. The advantage of passing an alphabetized list through each dictionary is speed. It has only to make a single pass through each dictionary.

AlphaSpell’s speed is further increased with the aid of index files that let AlphaSpell skip ahead through each dictionary. With the aid of index files and some other optimizations in the searching algorithm, AlphaSpell is even fast enough for interactive spell checking. That is, you can set up a text editor to call AlphaSpell for each word you type, and it will check the word so quickly it won’t slow down your typing. AlphaSpell comes with an ARexx script you can adapt for use with your text editor, and XES already makes interactive spell checking available for XDME.

AlphaSpell will also do much more besides spell checking. It has four different ways to help you find the correct spellings of misspelled words. It can measure rough phonetic similarities between words. It can measure the edit distance between words, which is the least number of insertions, deletions, and transpositions it takes to change one word into another. It can match wildcard patterns, and it can list anagrams. These last two features are also useful for answering crossword puzzles and scrambled word games.

AlphaSpell gives you full control over creating and maintaining dictionaries. You can take the union, the difference, or the intersection of two dictionaries. You can delete words from a dictionary through pattern matching. You can even generate entirely new dictionaries out of the most frequent words found in text files. This feature can help you generate dictionaries for languages AlphaSpell doesn’t support yet. Currently, AlphaSpell dictionaries are available for eleven languages: Afrikaans, Danish, Dutch, English, French, German, Icelandic, Latin, Norwegian, Spanish, and Swedish.

With version 5, AlphaSpell is shareware rather than freeware. This version is much better than the last version and finally worth paying something for. It uses a better compression format for the dictionaries, it uses proven guessing techniques, and it has the new ability to pattern match. Its dictionaries are cleaner than those in previous versions, and it is documented much better.

1.3 Features

```
Spell Checking
    An entire document
    Interactively as you type
Checking for misused words
Guessing words
Pattern matching
```

```
        Dictionary maintenance
        Counting words
        Building lists of words
        Listing Anagrams
```

1.4 Spell Checking

Spell checking is the principal thing that AlphaSpell does. It can quickly spell check an entire file, and with a text editor that is customizable enough, it can provide transparent interactive spell checking. It's designed in such a way that it does both sorts of spell checking in a flash. When it spell checks an entire document, it first sorts the words, then checks them against the dictionaries in order, so that it can begin each new search without covering old territory. By quickly jumping ahead to the words beginning with the first letter of the word it is searching for, and by taking advantage of the dictionary layout to quickly skim past words that don't match, AlphaSpell can check for a word in the blink of an eye. This great speed makes it suitable for interactive spell checking.

```
SEE ALSO:   The S Command
            The T Command
            Spell checking a document
            Interactive Spell Checking
```

1.5 Spell checking a document

To spell check a document with AlphaSpell, use the S command and put the file name after the command and before the dictionaries. It will list any words it doesn't find. You can then search for these words in the document and correct those that need correcting.

It's useful to do spell checking in conjunction with a text editor and an AlphaSpell GUI for that editor. For example, the GUI I've written for XDME puts the list of unfound words into a listview and lets you find, replace, and learn words from this list. If you don't want to use XDME, you can design a GUI for your text editor of choice, perhaps using my GUI as a model. You may be able to get a free registered version of AlphaSpell by writing a GUI for it.

```
SEE ALSO:   The S Command
```

1.6 Interactive Spell Checking

AlphaSpell can check words so quickly, you can use it for interactive spell checking without slowing down your typing. What is required is a text editor that will let you remap the space bar to run AlphaSpell on the last word you typed. An editor with ARexx and the ability to remap all the keys should be sufficient for this. I have begun to use AlphaSpell for interactive spell checking with XDME. Let me explain how interactive spell checking works with XDME. Even if you don't use XDME, this explanation can help you adapt another text editor for interactive spell checking with AlphaSpell.

I have mapped the space bar to read the last word typed, which it then sends to AlphaSpell with the T command. What is usually sent to AlphaSpell will just be a word. But sometimes it will be a string with punctuation or special characters. So when AlphaSpell receives the string, it removes extraneous characters from both ends, so that the first letter is always a letter, an apostrophe, or a digit, and the last character is always a letter. When it is finished checking for the word, it prints the word to standard output, and it puts a "1" or a "0" in the file indicated by the -o option. I have AlphaSpell send this value and the word to separate environment variables. I use "0" and "1" so that XDME can use this value in a conditional. A "1" indicates that it found the word, and a "0" indicates that it did not. When a word isn't found, XDME calls an ARexx script that produces a display beep, and it indicates in the titlebar what word wasn't found. To speed up the interactive spell checking, XDME copies the dictionaries and AlphaSpell to the RAM disk. When you turn interactive spell checking off, it deletes these from RAM.

SEE ALSO: The T Command

1.7 Finding commonly misused words with AlphaSpell

Their are just sum errors spell checking won't catch. Four example, homonyms, words witch sound alike, will knot be picked bye a spelling checker. But AlphaSpell can still help you with such words. If you create a list of words you are prone to misuse, you can use AlphaSpell's F command to check a document for them. This command works just like spell checking, except that it lists the words it does find rather than the ones it doesn't.

SEE ALSO: The F Command

1.8 Guessing words

To have AlphaSpell guess the correct spelling of a word, use the G command. Use the -w option to indicate the word.

AlphaSpell uses two different methods for guessing words. The default method is to use an algorithm based on the SoundEx algorithm. What this does is transform each string into a code that sort of represents the way it sounds. I say "sort of," because this algorithm is designed for use with English words, and the same letters often make different sounds in different English words. For example, "gh" can make a hard G sound, as in ghost, an F sound, as in rough, or no sound at all, as in through. This method prints a word whenever its code matches the code for the word given to the -w option. Since the SoundEx algorithm is based on the way letters sound in English, it may be less effective for other languages. It is most useful when you make spelling mistakes because you tend to spell words as they sound to you. It is not quite as effective at helping you fix typos.

The second method is much more reliable, and it is just as reliable for other languages as it is for English. This method checks whether the edit distance between two words exceeds a certain value. You specify this value with the -n option. The edit distance between two words is the least number of deletions, insertions, and transpositions it takes to change one word into another. This value will be zero when the two words are the same, and in the worst case, when the two words share no letters in common, it will be the sum of their lengths.

I recommend a low value for the -n option when you want AlphaSpell to guess a short word, and a larger value when you want AlphaSpell to guess a longer word. For short words, large values will often list too many words, and for long words, small values may yield no words.

In the unregistered version, i.e. without the keyfile, you can set the edit distance no higher than two.

```
SEE ALSO:   The G Command
            The -n Option
            On the Edit Distance Algorithm
```

1.9 On the Edit Distance Algorithm

I came across the edit distance method in a book by Graham A Stephen called _String Searching Algorithms_. I initially copied the Wagner-Fischer algorithm for calculating the Levenshtein distance between strings, which I then modified to handle transpositions and to calculate the edit distance instead. The run time of this algorithm was O(mn), where m and n are the string's lengths. In English, the run time was the product of the string's lengths. I then reworked it until I got an algorithm whose run time was approximately O(k), where k is the maximum edit distance between words. I also decreased the time it took to complete each major step. The original algorithm created a table and calculated each value of the table by calculating four different values and taking their minimum. I changed it to determine which value would be the minimum before actually calculating any of the values. My algorithm takes only O(k), because it doesn't reconstruct the table for each word. It uses what it can from the last table created, and it stops when it can determine that the edit distance will be greater than k.

```
SEE ALSO:   The G Command
            Guessing words
            The Levenshtein Distance
```

1.10 The Levenshtein distance

The Levenshtein distance between two strings is the minimum number of insertions, deletions, and substitutions it would take to transform one into the other. This differs from the edit distance by counting substitutions the same as insertions and deletions. For the edit distance, substitutions have a weight of two, because a substitution is equivalent to an insertion and a deletion. But for the Levenshtein distance, substitutions have a weight of one. I chose to use the edit distance rather than the Levenshtein distance, because I didn’t want words such as "cat" and "dog" to have a distance of only three. That would cause AlphaSpell to guess too many words. Both measures can be modified to handle transpositions, as I did for the edit distance.

```
SEE ALSO:   The G Command
            Guessing words
            On the Edit Distance Algorithm
```

1.11 Pattern Matching with AlphaSpell

With the P command, AlphaSpell will list for you each word from the dictionary matching the pattern designated with the -w option. AlphaSpell uses some pattern matching routines written and put into the public domain by John Kercheval. What follows is an excerpt from his documentation, reformatted and spell checked by me.

```
                     REGEX Globber (Wild Card Matching)

                A *IX SH style pattern matcher written in C
                    V1.10 Dedicated to the Public Domain

                             March  12, 1991
                               J. Kercheval
                       [72450,3702] -- johnk@wrq.com
```

```
*IX SH style Regular Expressions
================================
```

The *IX command SH is a working shell similar in feel to the MSDOS shell COMMAND.COM. In point of fact much of what we see in our familiar DOS PROMPT was gleaned from the early UNIX shells available for many of machines the people involved in the computing arena had at the time of the development of DOS and it’s much maligned precursor CP/M (although the UNIX shells were and are much more flexible and powerful then those on the current flock of micro machines). The designers of DOS and CP/M did some fairly strange things with their command processor and OS. One of those things was to only selectively adopt the regular expressions allowed within the *IX shells. Only ’?’ and ’*’ were allowed in filenames and even with these the ’*’ was allowed only at the end of a pattern and in fact when used to specify the filename the ’*’ did not apply to extension. This gave rise to the all too common expression "*.*".

REGEX Globber is a SH pattern matcher. This allows such specifications as *75.zip or * (equivalent to *.* in DOS lingo). Expressions such as [a-e]*t would fit the name "apple.crt" or "catspaw.bat" or "elegant". This allows considerably wider flexibility in file specification, general parsing or any other circumstance in which this type of pattern matching is wanted.

A match would mean that the entire string TEXT is used up in matching the

```
PATTERN and conversely the matched TEXT uses up the entire PATTERN.

In the specified pattern string:
     `*' matches any sequence of characters (zero or more)
     `?' matches any character
     `\' suppresses syntactic significance of a special character
     [SET] matches any character in the specified set,
     [!SET] or [^SET] matches any character not in the specified set.

A set is composed of characters or ranges; a range  looks  like  'character
hyphen character' (as in 0-9 or A-Z).  [0-9a-zA-Z_] is the minimal  set  of
characters allowed in the [..]  pattern  construct.  Other  characters  are
allowed (ie. 8 bit characters) if your system will support them (it  almost
certainly will).

To suppress the special syntactic significance of any  of  `[]*?!^-\',  and
match the character exactly, precede it with a `\'.

SEE ALSO:   The P Command
```

1.12 Dictionary Maintenance

```
        AlphaSpell  offers  a  variety  of  features  for  creating  and
maintaining dictionaries. For starters, the -c option instructs  AlphaSpell
to compress its output. This option is needed whenever you mean to create a
dictionary  for  use  with  AlphaSpell.  AlphaSpell  lets  you  manipulate
dictionaries like sets. It will combine two dictionaries into their  union,
or their intersection, or the difference between the two. It will also  add
words to a dictionary from an unordered list.

SEE ALSO:   The D Command
            The I Command
            The L Command
            The U Command
            The -c Option
```

1.13 Counting words

```
        AlphaSpell will count the words in a file  according  to  the  same
rules it uses to recognize words when extracting them for  spell  checking.
It recognizes as a  word  any  string  of  two  or  more  alphanumerics  or
apostrophes, which ends  with  a  letter.  AlphaSpell  recognizes  all  the
letters in the ISO character set.

SEE ALSO:   The N Command
```

1.14 Building clean lists of real words from large documents

Since Random House and Webster haven’t made freely available word lists, and since it becomes tedious and time consuming to type in words from a dictionary, another method is needed for generating dictionary files. The best method available is to scan many files and count how many times each word in each file appears. The idea is that words which appear most frequently are correctly spelled words.

AlphaSpell has three commands to help you create dictionaries by this method. The first of these is the Q command. This command reads files and creates a frequency list. A frequency list has a word, a tab, and a decimal number on each line. The number indicates how many times the word was found. Because frequency lists can get huge, I added the M command for merging frequency lists together. Before merging files together, it is best to randomize the lines in them. This will make reading them much faster, as in minutes verses hours. I have written a separate program for this. Finally, the W command weeds through a frequency list, printing each word that has appeared at least as many times as you specify with the -n option. For this, it is important to use an ordered frequency list, as it is created by AlphaSpell.

TECHNICAL DETAILS

The M and Q commands load words into a binary tree, and that takes a long long time for large lists of already alphabetized words, but much less time for unordered lists. Both of these commands give ordered lists as output. The W command just prints words as it finds them and does not load them into a tree. So it is just as fast with ordered or unordered lists. It will generate ordered output only if it receives ordered input.

```
SEE ALSO:   The M Command
            The Q Command
            The W Command
```

1.15 Listing anagrams

An anagram is a word with exactly the same letters as another word. With the A command, AlphaSpell will search for all the anagrams of a word.

SEE ALSO: The A Command

1.16 Usage of AlphaSpell

With AlphaSpell V, AlphaSpell has a UNIX like interface. This means that it accepts options preceded by hyphens, and it can recognize them in any order. Options may appear anyplace and do not have to precede the command. Here is a quick synopsis:

```
Usage: AlphaSpell [-cv] [-d drawer] [-k keyfile] [-n integer]
[-o output] [-w word] <command> [files ...]
COMMANDS:
         -A        List Anagrams            -P        Pattern Match
         -D        List Difference          -Q        Write Frequency List
         -F        Find Words               -R        Register AlphaSpell
         -G        Guess Word               -S        Spell check
         -I        List Intersection        -T        Test Word
         -L        Learn word               -U        List Union
         -M        Merge Frequency Lists    -W        Weed Frequency List
         -N        Count Words

OPTIONS
  -c  [^MNQRT] Compress Output     -n  [W] Minimum Word Frequency
  -d  [AFGLPS] Dictionary Drawer   -o  [?] Output file
  -k  [^MNQRT] Key file            -v  [^?] Print version
  -n  [G] Maximum Edit Distance    -w  [ALPT] Word or Wildcard Pattern
```

A word on the options: After each option, there is a set representing the commands each option works with. These sets are in the same format as those that AlphaSpell reads with the P command. Each set is expressed as briefly as I could put it.

1.17 The -A Command

```
Usage: AlphaSpell -A -w word [-d drawer] [-o output] dictionaries ...
```

This command tells AlphaSpell to list every anagram it finds of the word specified with the -w option. List all dictionaries after the options and the command. Use the -o option to send the anagrams to a file. Please make sure that the output file is a different file than any of the dictionaries. Otherwise, it won’t work. No other options have an effect on this command.

An anagram is a word with exactly the same letters as another word. This command is useful mainly for cheating at scrambled word games, such as Jumble in the newspaper. It would also be useful for cheating at Scrabble, but using a computer while playing Scrabble might look fishy to some people.

This feature is really just a vestige of my old guessing routines, which checked, among other things, whether two words had the same letters. The new guessing routines are more reliable and no longer check for this. But I decided to leave this feature in, because it has its own uses.

EXAMPLE:

```
    AlphaSpell -Aw evil English.low
```

Lists the anagrams of evil, such as "live," "vile," and "evil," but not "lvei," because it’s not an English word.

```
SEE ALSO:   Listing anagrams
            The -w Option
            The -o Option
            Input Files
```

1.18 The -D Command

```
Usage: AlphaSpell -D [-c] [-o output] file1 file2

        This has AlphaSpell read two dictionaries as input  and  spits  out
their difference as output. The difference of the two dictionaries  is  all
the words from the first dictionary named minus  all  the  words  that  the
second dictionary has in common with it. The output file  should  be  named
with the -o option, and the two dictionaries to be  read  should  be  named
after the command. Any dictionaries named after these two will be  ignored.
The output file may be the same as the first dictionary.  This  makes  this
command useful for removing words from a dictionary. The  dictionaries  may
be compressed in AlphaSpell's special format or uncompressed.

EXAMPLE:

    AlphaSpell -D Spangalese Spangalese.mix -c -o Spangalese.low

    This creates a lowercase Spangalese dictionary by subtracting the mixed
case words in a Spangalese wordlist.

SEE ALSO:   Dictionary maintenance
            The -o Option
            Input Files
```

1.19 The -F Command

```
Usage: AlphaSpell -F -s source [-c] [-d drawer] [-o output] Dictionaries

        This command reads a file and  checks  the  words  in  it  against
the dictionaries. It lists all the words it does find.

EXAMPLE:

    AlphaSpell -s letter.txt -F Work:Dictionaries/confusing.low

    This checks whether letter.txt contains any of the words in
confusing.low, and it lists any it finds.

SEE ALSO:   Finding commonly misused words with AlphaSpell
            The -c Option
            The -o Option
```

1.20 The -G Command

```
Usage: AlphaSpell -G [-n maximum edit distance] [-d drawer] [-o output]
[-c] -w word Dictionaries ...

        This command lists approximate matches to the word specified by the
-w option. The value given to the -n option affects how AlphaSpell chooses
approximate matches. If no value is given to -n, or if a value of zero is
given to it, AlphaSpell prints words that may sound like the word given to
```

the -w option. If you give a value to -n greater than zero, AlphaSpell prints words that are close in spelling. The value given to -n represents the maximum edit distance between two words. The edit distance is the least number of insertions, deletions, and transpositions it takes to transform one word into another. A smaller value will yield fewer words than a greater value. A high enough value, such as 100, will yield every word in the dictionary.

EXAMPLES:

```
    AlphaSpell -Gw ghoti Data:Dictionaries/*.low -o matches
```

This uses the SoundEx method to search for words that might sound like ghoti, and it will list words such as goat and goatee, putting them in the file called "matches." It is not sophisticated enough to know that ghoti is a homonym for fish: touGH + wOmen + cauTIon.

```
    AlphaSpell -Gk 2 -w lase -d Data:Dictionaries/ *.low *.mix -o matches
```

This checks the word "lase" against the dictionaries in Data:Dictionaries named by the wildcard patterns. It writes to "matches" any words with an edit distance of two or less from "lase," words such as "lace," "laser," and "case."

```
SEE ALSO:   The -n Option
            Guessing Words
```

1.21 The -I Command

```
Usage: AlphaSpell -I [-c] [-o output] Dictionary1 Dictionary2
```

This command lists all the words two dictionaries share in common, i.e. the intersection of the two dictionaries. The output file should be named with the -o option, and the two dictionaries to be read should be named after the command. Any dictionaries named after these two will be ignored. The output file may be the same as the first dictionary. The dictionaries may be compressed in AlphaSpell’s special format or uncompressed.

EXAMPLE:

```
    AlphaSpell -I ukacd.low common.low -o common.low -c
```

This compares a list of common words, perhaps generated by tabulating word frequencies in files, with the UK Advanced Cryptics dictionary. The effect is to remove from common.low any words not found in the much larger dictionary.

```
SEE ALSO:   Dictionary maintenance
            The -o Option
            Input Files
```

1.22 The L Command

```
Usage: AlphaSpell [-c] [-o output] [-w word] L Wordlists ...

        With this command, AlphaSpell sorts and lists all the words sent to
it. The wordlists may be AlphaSpell dictionaries, or contain one  word  per
line. They do not have to be ordered, and  AlphaSpell  will  actually  read
them faster if they are unordered. The purpose of this command  is  to  add
words to a user dictionary. To do so, your user dictionary  should  be  the
output file, and it should be one of the input files.  This  feature  takes
only a moment with a small user dictionary, but it would be incredibly slow
with the main dictionary. To speed it up a bit, name any unordered lists of
words first. To add words  to  the  main  dictionary,  use  the  U  command
instead. To add a single word to a user dictionary, you may specify it with
the -w option.

EXAMPLES:

    AlphaSpell -c -L -w McDuff -o Names.mix Names.mix

    This adds the name "McDuff" to a mixed case dictionary containing
names.

    AlphaSpell -Lco User.low words User.low

    This adds the words in the wordlist "words" to a user dictionary.
```

1.23 The -M Command

```
Usage: AlphaSpell [-o output] -M Files ...

        Use this command to merge frequency lists into one frequency list.

EXAMPLES:

    AlphaSpell -M -o freqlist *

    This reads all the files in the current directory, which should all be
frequency lists, and merges them into one list called "freqlist."

    AlphaSpell -M -o masterlist masterlist words

    This adds the frequencies listed in the file "words" to those in the
master frequency list "masterlist."

SEE ALSO:   Building clean lists of real words from large documents
```

1.24 The -N Command

```
Usage: AlphaSpell -N [-o output] File

        With this command, AlphaSpell  counts  the  words  in  a  file  and
briefly  displays  the  count.  Its  output  is  designed  to  go  into  an
```

```
environment variable that can then be displayed in a titlebar.  It’s  meant
to be used from a text editor that can display information in the titlebar,
such as XDME.

EXAMPLE:

    AlphaSpell cats.txt -N -o ENV:count

    This counts the words in cats.txt and stores the total in the
environment variable "count."

SEE ALSO:   Counting Words
```

1.25 The -P Command

```
Usage: AlphaSpell -P [-c] [-d drawer] [-o output] [-w wildcard pattern]
Dictionaries ...

        With this command, AlphaSpell lists all  the  words  in  the  named
dictionaries matching the wildcard pattern.  This  is  useful  for  finding
words and for cleaning up the dictionary.

EXAMPLES:

    AlphaSpell -Pw [pP]*t -d $DDIR $Dict

    This lists all the words in the dictionaries beginning with the letter
p and ending with a lowercase t, such as pat, Pat, and proletariat. It uses
the environment variables created by the Install script. $DDIR holds the
name of the dictionary drawer, and $Dict holds the names of the
dictionaries.

    AlphaSpell -w ??* -o Spangalese -P Spangalese

    This removes solitary letters from a list of Spangalese words.

    AlphaSpell -Pw ???? -d $DDIR *.low

    This searches the lowercase dictionaries in the dictionary drawer for
all four letter words.

    AlphaSpell -P -w *[A-ZÀ-Þ]* -o Spangalese.mix Spangalese -c

    This creates a dictionary of mixed case Spangalese words from a
Spangalese wordlist.

SEE ALSO:   Pattern Matching
```

1.26 The -Q Command

```
Usage: AlphaSpell -Q [-o output] Files
```

```
        Reads multiple files and creates a list of all the words found with
a count after each word of how many times it was found.

EXAMPLE:

    AlphaSpell *.txt -o freqlist Q

    This builds a list of word frequencies from all the files matching the
*.txt wildcard pattern.

SEE ALSO:   Building clean lists of real words from large documents
```

1.27 The -R Command

```
Usage: AlphaSpell -o output R

        This command asks you questions and generates an  order  form  from
your answers. Use this when you decide to register AlphaSpell. Alternately,
use the registration script.

EXAMPLE:

    AlphaSpell -R -o ram:form

    This asks for registration information and creates a form on the RAM
disk.

SEE ALSO:   Why Register AlphaSpell?
            How to Register
```

1.28 The -S Command

```
Usage: AlphaSpell -S [-c] [-d drawer] [-o output] File Dictionaries ...

        With this command, AlphaSpell checks the words in  a  file  against
the dictionaries named and lists the words it didn’t find. This command  is
used for spell checking an entire document.

EXAMPLE:

    AlphaSpell -S -s letter.txt -o unfound -d $DDIR $Dict

    This spell checks "letter.txt" and lists unfound words in the file
"unfound." It uses the environment variables $DDIR and $Dict to name the
dictionary drawer and the dictionaries.

    AlphaSpell -Ss T:temp -o T:temp -d $DDIR *.low *.mix

    This spell checks a temporary file saved by a text editor and
overwrites it with a list of unfound words. This is an economical use of
the same file when the actual text is held in the buffer of a text editor.
It uses $DDIR to name the dictionary directory and wildcard patterns to
```

```
read all the dictionaries in the directory.

SEE ALSO:   Spell Checking
            Spell Checking a Document
```

1.29 The -T Command

```
Usage: AlphaSpell -T [-d drawer] [-o output] -w word Dictionaries ...

        With this command, AlphaSpell quickly checks whether the word given
with the -w option is in any of the dictionaries. If the word is not  legal
as entered, it strips it of any extraneous leading and trailing  characters
before it begins its search. It sends a "1" to the output file if the  word
was found and a "0" otherwise.  It  sends  the  word  it  searched  for  to
standard output. This command is designed to be used for interactive  spell
checking.

EXAMPLE:

    AlphaSpell -Tw bat -d $DDIR common.??[!x] User.??[!x] English.??[!x]

    This checks "bat" against the dictionaries in the dictionary directory,
checking a short dictionary of common words first. If it doesn't find the
word there, it checks the user dictionaries, and it checks the main
dictionary only if it still hasn't found the word. By checking shorter
dictionaries of more commonly used words first, it takes less time to check
for the word.

    AlphaSpell >ENV:word -o ENV:ok -Tw that, -d $DDIR $Dict

    Checks for the word "that" and stores it in the environment variable
$word. It stores a "1" in the environment variable $ok if it finds the
word. Otherwise, it stores a "0" there. It uses environment variables to
know what dictionaries to read. The $Dict environment variable should name
shorter dictionaries of common words first to minimize searching time.

SEE ALSO:   Spell Checking
            Interactive Spell Checking
```

1.30 The -U Command

```
Usage: AlphaSpell -U [-c] [-o output] Dictionary1 Dictionary2

        This command combines the words in both dictionaries into a  single
dictionary. In other words, it prints the union of  the  two  dictionaries.
This is useful for merging a user dictionary,  or  some  other  dictionary,
with the main dictionary. To do so, name the main dictionary as  the  first
input file and as the output file. The output file should be named with the
-o option, and the two dictionaries to be read should be  named  after  the
command. Any dictionaries named after these two will be ignored. The output
file may be the same as the  first  dictionary.  The  dictionaries  may  be
compressed in AlphaSpell's special format or uncompressed.
```

```
EXAMPLE:

    AlphaSpell -U English.low User.low -o English.low -c

    This adds the words in the lowercase user dictionary to the main
lowercase dictionary.

SEE ALSO:   Dictionary maintenance
            The -o Option
            Input Files
```

1.31 The -W Command

```
Usage: AlphaSpell [-n min frequency] [-c] [-o output] W File

        This command weeds through a frequency list, printing each word
whose frequency is at least as great as the value set with the -n option.

EXAMPLE:

    AlphaSpell -Wo common -n 5000 freqlist

    This reads freqlist, a list of word frequencies, and creates a list of
all the words that appeared at 5000 times.

SEE ALSO:   Building clean lists of real words from large documents
            The -n Option
```

1.32 The -c Option

This option tells AlphaSpell to compress its output. You should use this option whenever you are creating a dictionary that AlphaSpell will use for spell checking. This option is useful for any command that has AlphaSpell list words. Instead of listing one word to a line, it will write the words in the compressed dictionary format used by AlphaSpell. It will also create an index file for the dictionary it writes. For this purpose, it needs a file name, which you must specify with the -o option. If you do not use the -c option, you may let the output go to standard output.

This option is disabled in the registered version.

1.33 The -d Option

Use this option to specify the drawer the dictionaries are in. This option saves you from having to type in the complete path for each dictionary you name. Whenever you do type in the full path name for a dictionary, AlphaSpell will ignore this option for that dictionary. So you can still use dictionaries from different drawers. This option just makes it easier to use many dictionaries from the same drawer.

SEE ALSO: Input Files

1.34 The -k Option

This option lets you specify the name and path of the keyfile. If you don’t use the -k option, the default name for the keyfile is "S:Alpha-Key". This option lets you put the keyfile anywhere you want. Of course, you have to have a keyfile to make use of this option. You get the keyfile when you register AlphaSpell.

1.35 The -n Option

With the G command, use this option to specify the maximum edit distance for guessing. If you don’t use this option, or if you pass it a value of 0, AlphaSpell will use SoundEx matching. Otherwise, it will list all the words within the specified edit distance from the target word.

With the W command, use this option to specify the minimum word frequency. The W command will create a list of all the words in the frequency lists whose frequency meets the minimum.

1.36 The -o Option

Use this option to specify the output file. If you don’t use this option, the output will go to standard output. If you use the -c option, the -o option is needed, so that AlphaSpell will know what name to give the index file it creates for the compressed output. It is also important to use this option when you want the output to go to a file you also want to read input from. If you tried to use a redirection operator, AlphaSpell wouldn’t be able to read the file, and you would just lose what was in it. But by passing AlphaSpell a name with the -o option, AlphaSpell can make sure that it doesn’t write to the file until after it is finished reading from it. In that way, you can have AlphaSpell read from and write to the same file. This is useful when you want to modify a file rather than just create a new one.

1.37 The -s Option

Use this option to indicate the source file to be spell checked with S or F command. The purpose of this option is to let you specify the file name without indicating a path. This prevents AlphaSpell from assuming the file is in the dictionary drawer, which is not a place where you should keep text files.

1.38 The -v Option

This option just prints the version string and quits. It’s only real purpose is to prevent the compiler optimizations from discarding the version string as deadwood.

1.39 The -w Option

Use this option to specify a word or wildcard pattern. The T, A, G, and L commands each expect a word after this option. The P command expects a wildcard pattern. The value after the -w option isn’t used with any other upper case options.

1.40 Input Files

Any arguments that don’t belong to options and are not options themselves are understood by AlphaSpell to be file names or wildcard patterns meant to match file names. It doesn’t matter where any arguments in the command line go. File names may be interspersed among the options, placed before the options, after the options, or whatever. The only order that matters is among the file names themselves. File order is important mainly for the -T and -D commands. Since AlphaSpell reads files in the order they’re named, naming shorter common word dictionaries first can speed up the -T command for common words. This can make interactive spell checking much faster. The order can also affect the speed of the -S and -F commands, but not to the same degree as it affects the speed of the -T command. For the -D command, the order is essential, because it takes the asymmetric difference of two files. But for other commands, the order of the filenames never affects anything more than the order of the output, and for some commands, it doesn’t even affect that.

If a directory is specified with the -d option, any files named without a path are searched for in the directory specified by the -d option.

It should be perfectly safe to make any of the input files the same as the output file. When they are the same, AlphaSpell will make sure to write to the file only after it has finished reading it.

SEE ALSO: The -d Option

1.41 Legal Matters

Copyright
Distribution
Disclaimer
Legal Use
Registering

1.42 Copyright

AlphaSpell Copyright © 1995 Fergus Duniho

You are NOT ALLOWED to modify AlphaSpell V or the documentation in any way. Packing and archiving do not count as modifications. You are NOT allowed to decompile AlphaSpell. This copyright does not extend to the words in the dictionaries. You may use them in any way you please, as Humpty Dumpty advocates, though others have questioned the wisdom of this approach.

This copyright protection does not apply to the previous version of AlphaSpell, version 4.00, which was copylefted under the GPL. Version 5.00 of AlphaSpell is not under the GPL. It is shareware.

1.43 Distribution

The unregistered version of AlphaSpell is freely distributable by any normal electronic means. Anyone may distribute it so long as they keep the entire contents of this package intact and unchanged. This package contains the following: AlphaSpell, AlphaSpell.guide, AlphaSpell.gui, Dict.low, Dict.ldx, Dict.mix, Dict.mdx, Install, and Register.

AlphaSpell may NOT be distributed by anyone whose advertising is likely to mislead people into believing that AlphaSpell is public domain. If anyone advertises that AlphaSpell is available on one of the disks they distribute, they must briefly explain what shareware is and make it clear that some of the disks they distribute contain shareware. Otherwise, they may not distribute AlphaSpell.

If you got AlphaSpell from a disk that you paid money for, you still haven't paid for AlphaSpell until you have paid me, Fergus Duniho, the registration fee. What you paid money for was the disk the program came on, not the program itself. If you decide to continue using AlphaSpell after a period of one month, you must pay me the registration fee.

The registered version of AlphaSpell may not be distributed by anyone by any means. Re-distributing the registered version is software piracy.

1.44 Disclaimer

By using this product, you accept the FULL responsibility for any damage or loss that might occur through its use or the inability to use it. The author of this program can NOT be held responsible.

Furthermore, I do not gaurantee that any AlphaSpell dictionary is complete or 100% accurate. I cannot be held liable for the inaccuracy or incompleteness of any AlphaSpell dictionary. I cannot even be held liable when an AlphaSpell dictionary is a gross misrepresentation of the language it is supposed to be a dictionary for. This is actually a real possibility, as I could mistranslate a character set used in a wordlist for a language I don't know. So be on your gaurd.

Moreover, I cannot be held liable for the presence of offensive words in any AlphaSpell lexicon.

1.45 Legal Use

You may use AlphaSpell free for a trial period of one month. If you decide to continue using it after that time, you must pay the shareware fee. This fee is $20.00 in United States currency or $30.00 in Canadian currency. An alternative to paying the shareware fee is to help me expand the market for AlphaSpell. You can do this by providing me with an AlphaSpell GUI for a text editor that doesn't yet have one, by providing me with a dictionary for a language AlphaSpell doesn't yet support, or by translating the documentation into another language.

1.46 The GUI for XDME

This section will be useful whether you want to use AlphaSpell with XDME or design a GUI for some other text editor or word processor. For those who use XDME, this section will tell you how to use the GUI I have designed for it. For those who want to design a GUI for some other editor, this section will give you ideas and a model to go by. If you plan on designing another GUI, be sure to look at AlphaSpell.gui with GadToolsBox.

The AlphaSpell GUI for XDME is a GadToolsBox GUI. XDME has the ability to use such GUI's asynchronously. The AlphaSpell GUI contains the following gadgets:

```
    A Cycle Gadget
    A Listview
    A String Gadget
    A Slider Gadget
    Ten Button:
        Clear
        Save
        Learn
        Discard
        Find
        Replace
        Edit
        Anagrams
        Match
        Guess
```

1.47 The Lists

The AlphaSpell GUI maintains three lists. These are:

The List of Unfound Words

```
        The List of Guesses, Matches, and Anagrams
        The List of Words to Learn
```

It displays each list in the same listview, and the cycle gadget cycles between the lists.

1.48 The List of Unfound Words

When AlphaSpell finishes spell checking a document, the GUI pops up with this list in the listview. This is the list of all the words AlphaSpell didn’t find in any of the dictionaries. The GUI lets you search for these words, and it lets you replace those that need correcting. It will help you find the correct spelling of a word when you don’t know it yourself.

1.49 The List of Guesses, Matches, and Anagrams

When AlphaSpell guesses at a word, matches a pattern, or lists the anagrams of a word, the results go into this list. When this list is on display, clicking on a word will automatically put it into the string gadget. With the other lists, you would have to click on the edit button first.

1.50 The List of Words to Learn

Whenever you choose to add a word from the first list to the user dictionary, it goes to this list. You can send the words in this list to the user dictionary with the "Save" button. Before saving the words, you have the opportunity to edit them in the string gadget. You can move words back to the list of unfound words with the "Discard" button.

```
SEE ALSO:   The Edit Button
            The Save Button
            The Learn Button
            The L Command
```

1.51 The Cycle Gadget

The cycle gadget determines which of the three lists appears in the list view.

```
SEE ALSO:   The Lists
            The Listview
```

1.52 The Listview

The listview shows one of three lists. Right after spell checking, it displays the list of unfound words. This is the list of all the words from your document that AlphaSpell didn’t find in any of the dictionaries you checked. If you ask it guess at a word, to match a pattern, or to list anagrams, it will put the results in list two and immediately display that list. The third list is for setting aside words from the first list, either for saving them or for discarding them. You can switch between the lists with the cycle gadget.

```
SEE ALSO:   The Lists
            The Cycle Gadget
            The String Gadget
```

1.53 The String Gadget

The string in the string gadget is used by the replace button when you replace a misspelled word in your document. If you know the correct spelling, just put the word in the string gadget with the edit button and edit it. The find button will use the word in the listview, and the replace button will use the word in the string gadget. If you don’t know the correct spelling, put the word there anyway and use the guess, match, or anagrams button to find the correct spelling. These buttons use the string in the string gadget, not the word displayed in the listview. When you’re in the list of words to learn, editing in the string gadget changes the word currently displayed in the listview.

```
SEE ALSO:   The Listview
            The Replace Button
            The Edit Button
            The Anagrams Button
            The Match Button
            The Guess Button
            The -w Option
```

1.54 The Slider Gadget

The slider gadget controls the maximum edit distance that AlphaSpell uses when guessing. This is the value that AlphaSpell reads with the -n option, and it affects nothing but guessing. A value of zero tells AlphaSpell to guess by rough phonetic similarity, and any other value tells AlphaSpell to find all the words within a certain edit distance from the search word.

```
SEE ALSO: The Guess Button
            The -n Option
            Guessing words
            On the Edit Distance Algorithm
```

1.55 The Clear List Button

This button clears the list currently on display. Use it wisely, because you can’t get a list back except by repeating the steps that created it.

```
SEE ALSO:   The Lists
```

1.56 The Save Button

When you press this button while the list of words to learn is on display, it will save the words in that list to the user dictionary, and it will then clear the list. Otherwise, it does nothing.

```
SEE ALSO:   The List of Words to Learn
            The Learn Button
            The L Command
```

1.57 The Learn Button

When the list of unfound words is on display, this button transfers the current word in that list to the list of words to learn.

```
SEE ALSO:   The List of Words to Learn
            The Save Button
            The L Command
```

1.58 The Discard Button

When you press this button while the list of words to learn is on display, it returns the word to the list of unfound words. Otherwise, it just removes the word altogether. Going through the list of unfound words is easier when you discard the misspelled words as you take care of them. This is because the listview has the annoying habit of going back to the top when you cycle between lists. If you go from top to bottom and eliminate words as you go along, the words you want to deal with will always be at the top of the listview.

1.59 The Find Button

This button searches for the next occurrence of the current word in the list of unfound words. When the last search was for another word, it begins its search from the top. Otherwise, it begins its search from wherever the cursor is. The searching it does is whole word searching. Use the find button in conjunction with the replace button to find and replace misspellings in the text.

```
SEE ALSO:   The Replace Button
            The List of Unfound Words
```

1.60 The Replace Button

After you find a word with the find button, use this button to replace it with its correction. The correction should be in the string gadget. You can type it there yourself, or you can ask AlphaSpell to find it for you with the guess, match, or anagrams button.

```
SEE ALSO:    The Find Button
             The Edit Button
             The Anagrams Button
             The Match Button
             The Guess Button
```

1.61 The Edit Button

This button copies the current word in the listview to the string gadget. In the list of unfound words, this is useful for edit a replacement string to use with the replace button. It is also required for using the guess, match, or anagrams buttons. These buttons use the string gadget string. In the list of words to learn, editing the current word changes it in the list. In the list of guesses, matches, and anagrams, this button serves, because clicking on a word automatically puts it in the string gadget.

```
SEE ALSO:   The String Gadget
            The Replace Button
            The Anagrams Button
            The Match Button
            The Guess Button
```

1.62 The Guess Button

This button tells AlphaSpell to guess at the word in the string gadget. It uses the value of the slider gadget for the edit distance. The result goes to list two, replacing what is already in it.

```
SEE ALSO: The Slider Gadget
            The String Gadget
            The G Command
            Guessing words
            On the Edit Distance Algorithm
            The Edit Button
            The List of Guesses, Matches, and Anagrams
```

1.63 The Match Button

This button tells AlphaSpell to list all the matches it finds of the wildcard pattern in the string gadget. This will not work without a valid wildcard pattern. The result goes to list two, replacing whatever is already in it.

```
SEE ALSO:   The String Gadget
            The P Command
            Pattern Matching
            The Edit Button
            The List of Guesses, Matches, and Anagrams
```

1.64 The Anagrams Button

This button tells AlphaSpell to list all the anagrams of the word in the string gadget. The result goes to list two, replacing what is already in it.

```
SEE ALSO:   The String Gadget
            The A Command
            Listing Anagrams
            The Edit Button
            The List of Guesses, Matches, and Anagrams
```

1.65 Why register AlphaSpell?

```
        Doing the right thing matters to me.
        I don’t care what’s right or wrong.
        What you get for registering.

SEE ALSO:   How to Register AlphaSpell
```

1.66 Moral reasons for registering

```
        The Golden Rule
        Objectivism
        The Categorical Imperative
        Universal Prescriptivism

SEE ALSO: How to Register AlphaSpell
```

1.67 The Golden Rule

If you wrote some shareware, would you want someone who could afford to pay for it not to pay for it? If not, please do as you would have done to you and pay for the shareware you use.

1.68 Objectivism

If you use this program regularly with no intention to pay for it, that is a violation of my property rights. To violate property rights is to live a secondhand existence and to ask people to live for your sake.

1.69 The Categorical Imperative

One formulation of the categorical imperative tells us to treat others as ends and never as means only. If you use shareware with no intention of paying for it, you are treating the author of the shareware as nothing but a means to your own well-being and productivity. But if you pay the authors of shareware you use, you are treating them not only as means but as ends.

1.70 Universal Prescriptivism

Consider the principle that it is morally permissible to use shareware without paying for it even when you can afford to pay for it. A few of you may follow this principle. But supposing you do, would you be willing to prescribe that everyone follow it? Do you think the world would be a better place if people regularly paid for shareware, or if people regularly used shareware without paying for it?

Certainly, someone may say, "There is just so much more good software available for me to use when I follow the principle of not paying for shareware." But would this still be true in a world in which no one paid for shareware? If no one paid for shareware, some people would still distribute freeware. But the incentive to write shareware would be gone. People who wanted to earn a living through programming would have to turn to commercial software instead. Commercial software would cost more, and it wouldn’t be freely distributable. So if no one paid for shareware, less software would be freely distributable.

Of course, someone may reply, "So what? The world isn’t like that. People do pay for shareware, and I’m not going to change that by not paying for shareware myself." And it is indeed true that other people will still pay for shareware even if some people refuse to. But those who do are taking advantage of a system whose existence requires that most people do not act like them. The shareware they use is available, because other honest people do pay for shareware. Such people leech off of a system that they don’t maintain, and so act immorally.

1.71 Morality is for suckers

And now for a different view on shareware. For this opinion, I interviewed a big fat cat who named his dog after a tool of destruction, his daughter after a firearm, and his son after himself. I have withheld his name to avoid embarrassing the company he is associated with.

"Shareware authors are saps. They think people will pay them for stuff they can just take for free. I just use other people's shareware and laugh at how stupid they are to make it freely distributable."

But the registered version is better. So you might want the registered version anyway.

"I'll just pirate it when I can. Some goof is bound to actually pay for it."

But if everyone behaved like you, no one would register it.

"What kind of fool do you take me for? I don't want other people to behave like me, you dipstick! I like being smarter than everyone else. That's how I get ahead!"

So you're saying it's stupid to be moral?

"That's right. Morality is for suckers. Anyone with half a brain is out for #1."

1.72 What you get for registering

What I send you
What else you get

1.73 What I send you when you register

When you register AlphaSpell, you will receive a pseudo keyfile. AlphaSpell comes with a program for translating the pseudo keyfile into the real keyfile. I distribute the pseudo keyfile rather than the actual keyfile only because the pseudo keyfile is easier to transfer. The keyfile is binary, whereas the pseudo keyfile is ASCII. I will send you the pseudo keyfile by email, and I will send you a printout of the pseudo keyfile by regular mail. The program for creating the keyfile will work either by reading the pseudo keyfile or by reading in the numbers of the pseudo keyfile as you type them in. Each number in the pseudo keyfile is between 0 and 255 and corresponds to a character in the keyfile. You can generate the keyfile from the pseudo keyfile with the keyconv program. Redirect keyconv's output to S:Alpha-Key, and send the pseudo keyfile as its input, typing a line such as:

```
keyconv >S:Alpha-Key <pseudokey
```

If I can't send you the pseudo keyfile by email, just type "keyconv >S:Alpha-Key" and type in the numbers from the pseudo keyfile into the program. Press control-\ when you are finished.

With the keyfile, you will have full access to all of AlphaSpell's features. In AlphaSpell V, the keyfile is required for compressing the output, and it is required for setting the edit distance greater than two when you want it to guess a word. These features are disabled in the unregistered version to remind you that you are using an unregistered version and to give you incentive to register. This is not because I think you are dishonest, but because even honest people want to get something more when they register shareware.

AlphaSpell will continue to use the same keyfile as long as I continue to work on it. So once you have the keyfile, you can use all the features in any future version of AlphaSpell.

1.74 What else you get for registering

When you register AlphaSpell, you not only get the keyfile; you also encourage me to put more work into AlphaSpell, making sure that it is a quality product that meets your needs. When I write Freeware, I write it mainly for myself, I don't test it thoroughly, and and I don't put as much effort into making sure it is a quality product. Take XES, for example. It makes XDME a lot easier for me to use, especially since I wrote it and know what everything does, but it isn't documented fully, and you will have to study it at the source code level to get the most out of it. Or take the DDLI. I haven't updated that in a long time. People seem to like it, but I could devote more time to it if I wished and make it even better. I just work on it when the mood strikes me, and although I've modified it beyond the latest release, I haven't gotten around to releasing the new version.

But things are altogether different with AlphaSpell. This is shareware. So I try to make it as good a product as I can, because I want you to use it and pay me for it. I'm interested in propagating its use around the globe. So I've made around a dozen AlphaSpell dictionaries for different languages, most of which I'll never need myself. When you pay for AlphaSpell, you will be telling me that my efforts are well directed, that I should continue making AlphaSpell better and better, and that I should continue to make sure that AlphaSpell is the best spelling checker you could use.

1.75 How to Register AlphaSpell

To register AlphaSpell send $20.00 to

Fergus Duniho
1095 Genesee St.
Rochester, NY 14611-4148

Please send $20.00 in American currency or from a check drawn from an American bank. Please include both a mailing address and an email address. I will send you the keyfile script through both email and regular mail. To create a registration form, use the registration script or type "AlphaSpell -o form R".

An alternate way to register AlphaSpell, i.e. an alternative to giving me money, is to design a GUI for using AlphaSpell with a text editor that doesn't already have an AlphaSpell GUI available for it. I have designed one for XDME, but all other Amiga text editors are up for grabs. In this guide, I have described the GUI for XDME, and I recommend using it as a model for other GUI's. If there is another text editor that supports *.gui files, you can even use the AlphaSpell.gui file I designed for AlphaSpell. The conditions for getting a registered copy of AlphaSpell are that you make the GUI freeware, not shareware, that you give me the right to distribute it with AlphaSpell, and that I test it with the editor it is for and find that it works. If someone has already done a GUI for your editor, you can still get a registered copy by doing a better GUI for the same editor. For XDME, a better GUI is one I would rather use than my own.

SEE ALSO: The GUI for XDME

1.76 About the Author

Other software by Fergus Duniho include XES and the DDLI. Both are on the Aminet. XES includes a GUI for using AlphaSpell with XDME.

You can send me email at fdnh@troi.cc.rochester.edu. If Counsellor Troi isn't your favorite character on Star Trek: The Next Generation, you can also send mail to any of the following addresses:

```
fdnh@ro.cc.rochester.edu
fdnh@picard.cc.rochester.edu
fdnh@riker.cc.rochester.edu
fd001d@uhura.cc.rochester.edu
```

Since Ro was my favorite character on the show, I often log on to Ro rather than Troi, but all my email gets put on the same mail spool anyway.

My mailing address is

Fergus Duniho
1095 Genesee St.
Rochester, NY 14611-4148
USA

I expect it will be good for a couple more years, and I expect my current email addresses will be good for the same amount of time. If you want to make sure you have my most current address, check the author information on the DDLI Page. This is a World Wide Web page devoted to another program of mine, called the Duniho and Duniho Life Pattern Indicator. Since this page is not in my own personal account, its location does not depend on my email address. It is listed in Yahoo, and its current URL is:

```
http://sunsite.unc.edu/pub/academic/psychology/
alt.psychology.personality/html/ddli.html
```

If no new address is listed there for me there, I haven't moved yet. When I do move, I will put my new address there. This page will always contain a mailto: link to my current email address.

1.77 History

Ancient History
Revisions since 5.0

1.78 The Ancient History of AlphaSpell

AlphaSpell has its roots in a spelling checker I wrote in ARexx back in Fall 1991 or Spring 1992. I learned C in the Spring of 1992 and rewrote that spelling checker in C for my first major project in C. Initially, it was mainly a brute force spelling checker. Its main virtue was that it finished spell checking quickly, because it spell checked words in alphabetical order, thereby requiring only one pass through the dictionary.

AlphaSpell V2.00 was also in C, but I never released it. It had the ability to use a compacted dictionary.

AlphaSpell V3.00 was a new C++ program. It implemented the same basic algorithm as V2.00, but with less redundancy. This version also did the tasks that previous versions depended on other programs to do. Previous versions required other programs to get the words from a file, to sort those words, and to remove redundancies, in order to create an alphabetized list that AlphaSpell could read. AlphaSpell V3.00 did all this on its own.

AlphaSpell 4.00 is in C again, because C++ has changed, and some of my old C++ code is broken. Unfortunately, I don’t have references on the latest revision of the language. So I translated it all back into C and added a bunch of new features. New features include word counting, guessing, testing, weeding, and the ability to work with multiple dictionaries, both compressed and uncompressed. Previous versions expected input from standard input. Version 4.00 does not unless you tell it to by using "stdin" as a file name.

AlphaSpell 5.0 is a C++ program again, and the code is just about a complete reworking of AlphaSpell 4.00’s code. New features include a UNIX like interface, new guessing algorithms, a new compression format for dictionaries, pattern matching, and a new GUI for XDME. The new guessing algorithms are based on sounder methods than I employed in the last version. One measures phonetic similarity between words, and the other checks how easily one word can be changed into another.

1.79 Revisions of AlphaSpell since 5.0

```
ABBREVIATIONS: BF = Bug Fix, NF = New Feature, CF = Changed Feature
               CM = Code Modification, OP = Optimization
```

AlphaSpell 5.8 Saturday, 2 September 1995
AlphaSpell 5.7 Sunday, 27 August 1995 (Unreleased)
AlphaSpell 5.6 Saturday, 26 August 1995
AlphaSpell 5.5 Friday, 25 August 1995

```
        AlphaSpell 5.4 Tuesday, 22 August 1995 (Unreleased)
        AlphaSpell 5.3 Sunday, 20 August 1995
        AlphaSpell 5.2 Sunday, 20 August 1995 (Stopped in transit)
        AlphaSpell 5.1 Thursday, 17 August 1995
```

1.80 AlphaSpell 5.8 - 2 September 1995

```
OP - I wrote an optimally fast strcmp in assembly and replaced cmpstr with
     it. So string comparisons are now as fast as I can make them. As with
     cmpstr, this strcmp treats characters as unsigned.
```

1.81 AlphaSpell 5.7 - 27 August 1995

```
NF - The -U, -I, and -D commands can now handle single files. When a single
     input file is passed to one of these commands, it acts as though a
     second empty file has also been passed to it. The -U command lists the
     union of the one source file and the empty file, which is just the
     words in the source file itself. The -D command lists their
     difference, which is also all the words in the source file. The -I
     command lists their intersection, which is an empty list.

NF - If no input file is given, the -U, -I, -D, and -N commands read input
     from standard input. This is useful for piping a source file to
     AlphaSpell.

NF - If no source file is named with the -s option, the -S and -F commands
     read input from standard input.

CM - To reduce code size, I removed the dump() and wrdcmp() functions,
     since they are no longer needed. Removing dump() makes -Pw * slightly
     slower, but -U or -D can now do what -Pw * did as fast as it did it.
```

1.82 AlphaSpell 5.6 - 26 August 1995

```
BF - The -T command in 5.5 was returning a "0" for words one character in
     length rather than a "1". This caused interactive spell checking to
     complain about all one letter words. This is now fixed. AlphaSpell now
     returns a "1" for any word one letter long.

BF - Changes made in 5.5 disabled word counting for files named as
     arguments. It worked only for standard input. That is now fixed.

OP - Faster reading of source files for spell checking.

OP - Faster string comparisons.
```

1.83 AlphaSpell 5.5 - 24 August 1995

```
CF - Commands replaced by command options. THIS CHANGE WILL BREAK ANY
     SCRIPTS THAT CALL ALPHASPELL. To accommodate this change, precede each
     command with a hyphen. Between this change and the new feature
     added in 5.4, arguments to AlphaSpell can now be entered in any order.

NF - Input files may now be identified by wildcard patterns. These wildcard
     patterns are the same as those recognized by the -P command. Matching
     works only for filenames, not for path names.

NF - A new option, the -s option, indicates the name of the file to be
     spell checked. THIS OPTION BREAKS OLD SCRIPTS. To fix them, include
     the -s option wherever you've used the -S or -F commands.

NF - The directory named by the -d option is now the default directory for
     any input file that doesn't have a path given for it. It does not
     affect the name given with the -s option. The -s option was created
     just to prevent this, since the -d option is generally for naming a
     drawer with multiple dictionaries in it.

NF - The output file may match any of the input files, including the file
     named with the -s option and files named by wildcard patterns. When
     necessary, AlphaSpell writes its output to a temporary file, which it
     copies to the intended output file after it has closed it for reading.

BF - When AlphaSpell extracts words from a file, it no longer recognizes
     any string beginning with a hyphen as a word.

BF - Spell checking broke in 5.3. It is now fixed.
```

1.84 AlphaSpell 5.4 - 22 August 1995 (Internal)

```
NF - I replaced Daniel Barrett's getopt() code with the GetOpt class in
     libg++. This allows options to appear any place on the command line.
     You are no longer restricted to placing all the options before the
     command.
```

1.85 AlphaSpell 5.3 - 20 August 1995

```
BF - Replaced strcmp() with a faster function that treats characters as
     unsigned: cmpstr().

BF - Recompiled everything with the -funsigned-char switch on.

CM - Changed any occurrence of "unsigned char" to "char".

OP - Removed some unneeded code from a comparison function: wrdcmp().

CM - Removed a no-longer-needed comparison function: dctcmp().

OP - Now compares common prefixes even faster than 5.2 does.
```

1.86 AlphaSpell 5.2 - 20 August 1995 (Stopped in transit)

```
OP - For some case insensitive comparisons, AlphaSpell creates a temporary
     lowercase string and then proceeds with a case sensitive comparison.
     This should speed it up, because case sensitive comparison is faster,
     and because it often uses the same temporary lowercase string more
     than once.

OP - AlphaSpell compares common prefixes faster.
```

1.87 AlphaSpell 5.1 - 17 August 1995

```
BF - AlphaSpell no longer stops prematurely when it reads a blank line.

BF - The A, G, and P commands can now use the first source file as the output
     file.

BF - When reading straight text files, AlphaSpell isn't limited to
     recognizing only those characters that can appear in legal words.
     Instead, it just reads whole lines. This is useful when you want to
     use pattern matching to check what characters a wordlist you found
     uses for special characters.
```

1.88 Credits and Acknowledgments

AlphaSpell is a program by Fergus Duniho. I compiled it with GCC and libnix. After unsuccessfully trying to add sets to a wildcard matching routine I wrote, I scrapped it in favor of J. Kercheval's SH style pattern matching routines. Kercheval's code is labeled as public domain, and it is part of Bob Stout's snippits collection, available in the SimTel Software Repository.

Thanks go to Michal Kara for DED (a.k.a. Disk-Editor) and to Rainer Koppler for Cvt. DED helped me recover one of the dictionaries and most of this file when I accidently overwrote them. Cvt helped me convert some wordlists I found to ISO Latin.

1.89 Dictionaries available for AlphaSpell

To date, AlphaSpell dictionaries are available for the following languages:

```
 Afrikaans    Danish       Dutch        English      French
 German       Icelandic    Latin        Norwegian    Spanish
 Swedish
```

Previously, this documentation said that an Italian dictionary would be available. Unfortunately, the Italian word list I have lacks accents. So I've chosen not to use it.

1.90 Afrikaans Dictionaries

```
FILE: Afrikaans.lha
SHORT: Afrikaans dictionary for AlphaSpell V
Type: text/edit
Uploader: fdnh@troi.cc.rochester.edu
Author: Fergus Duniho and Bernard Nieuwoudt
Version: 1.0

        This is an Afrikaans AlphaSpell dictionary based on:

        ftp://sable.ox.ac.uk/pub/wordlists/afrikaans/afr_dbf.zip

        Bernard Nieuwoudt, the author of the original file,  says  "If  the
user of the list makes alterations to the list, the list then  becomes  the
property  of  that  user.  I  then   relinquish   all   rights   (and   all
responsibility) of any sort to the list." Since I have made alterations  to
the original list, the ownership of  the  new  list  falls  to  me,  Fergus
Duniho. The original list contained words with spaces, such as  "a  priori"
and "a capella-koor," but the new list  does  not  contain  spaces  in  any
words. It contains "priori" and "capella-koor,"  which  the  original  list
doesn't. I made this alteration, because AlphaSpell recognizes the space as
a word delimiter. It can't tell you whether "a priori"  is  spelled  right,
but it can tell you whether "priori" is.

        What follows is the text from Bernard Nieuwoudt's  original  readme
file:

This is a message for the accompanying file AFR_DBF.ZIP:

That file contains a list of Afrikaans words. It is PKZIP-ed (PKZIP 2.04g)
and posted in binary from a DOS platform.

This list was compiled in personal capacity since 1984. Many works were
referenced, but the major part of the list was compiled by myself.

Users of the list agree, by using it, to the following:

1) The list is used entirely at own risk. I will not be held liable for any
   mistakes, ommisions, law suites etc.
2) The use of the list as it was posted, is for personal use only.
3) As there were many sources for the list, credit cannot be given to all
   sources. It is suggested that any commercial use of part of the list
   should first be vetted by lawers.
4) Commercial use of the list, as it is, should first be cleared out with me.
5) If the user of the list makes alterations to the list, the list then becomes
   the property of that user. I then relinquish all rights (and all
   responsibility) of any sort to the list.

I hope this makes sence and that you find it agreeable?

Greetings
Bernard Nieuwoudt

TEL: (012) 420 3637
EMAIL: BERNARD@CCNET.UP.AC.ZA
```

1.91 Danish Dictionaries

```
FILE: Danish.lha
SHORT: Danish dictionary for AlphaSpell V
Type: text/edit
Uploader: fdnh@troi.cc.rochester.edu
Author: Fergus Duniho and Unknown
Version: 1.0

        This is a Danish AlphaSpell dictionary based on:

        ftp://sable.ox.ac.uk/pub/wordlists/danish/danish.words.Z

        I don’t know who originally compiled  the  wordlist.  The  original
wordlist was entirely in 7-bit ASCII, using what I took to be  the  Denmark
II character set. So I wrote  a  conversion  script  for  Rainer  Koppler’s
program Cvt, which translates ASCII characters in the Denmark II  character
set to their 8-bit equivalents in ISO Latin. That  script  is  included  in
this package.

        Please note that the wordlist was  entirely  in  lowercase,  and  a
cursory examination of the file suggests that names have been  included  in
lowercase. For example, I saw the  string  "anna"  in  the  file.  So  this
package does not  include  a  mixed  case  dictionary  for  case  sensitive
checking.
```

1.92 Dutch Dictionaries

```
FILE: Dutch1.lha
SHORT: Dutch dictionary for AlphaSpell V
Type: text/edit
Uploader: fdnh@troi.cc.rochester.edu
Author: Fergus Duniho and Erik Frambach
Version: 1.0

    This dictionary is based  on  an  archive  I  found  on  SimTel  called
nlword10.zip. The original archive contains  26  files,  nl.a,  nl.b,  ...,
nl.z. These were plain text files using the IBM character set,  and  sorted
alphabetically. To  construct  this  dictionary,  I  removed  the  carriage
returns, converted the non-ascii characters from IBM to ISO,  resorted  the
files, and fed them all into a single file. The total number  of  words  in
the dictionary is approximately 220,000.

    If you find any mistakes in the dictionary, don’t write to me. I  don’t
know any Dutch. This dictionary is not maintained by  me.  It  based  on  a
dictionary maintained  by  Erik  Frambach.  Please  send  any  comments  on
misspelled or missing words to him. His address is:

E.H.M.Frambach@eco.rug.nl

    Here is what he wrote in his original readme file:

This Dutch dictionary contains 26 files, NL.A, NL.B, ..., NL.Z.
Each file contains words that start with the letter indicated by
```

```
the file extension.
The files are plain text files, the words are in extended ASCII,
each word is on a separate line, and they are sorted alfabetically
per file. The total number of words is approximately 220,000.

This is version 1.0 of the Dutch dictionary.
Please send any comments on misspelled or missing words to
E.H.M.Frambach@eco.rug.nl
```

1.93 English Dictionaries

```
FILE: ukacd.lha
SHORT: Big English dictionary for AlphaSpell V
Type: text/edit
Uploader: fdnh@troi.cc.rochester.edu
Author: Fergus Duniho and Ross Beresford
```

This dictionary is based on version 1.3 of Ross Beresford's "UK Advanced Cryptics Dictionary." This is "a word list for crossword solvers and setters." The main differences between Ross Beresford's original dictionary and the AlphaSpell dictionary are these. (1) The AlphaSpell dictionary is one file, and the original dictionary is 26 files. (2) The AlphaSpell dictionary is compressed, and the original dictionary is straight ASCII. (3) The original dictionary contained phrases as well as words, but the AlphaSpell dictionary does not contain the phrases. This is because AlphaSpell is primarily a spelling checker, and it only checks words against words, not against phrases.

Although AlphaSpell is primarily a spelling checker, it can also be used to solve crossword puzzles. For example, if you're looking for a five letter word whose second letter is p, you could send the pattern "?p???" to AlphaSpell. AlphaSpell is also useful for solving scrambled word puzzles.

1.94 French Dictionaries

```
FILE: French.lha
SHORT: French dictionary for AlphaSpell V
Type: text/edit
Uploader: fdnh@troi.cc.rochester.edu
Author: Fergus Duniho and Unknown
Version: 1.0
```

This AlphaSpell dictionary is based on the French dictionary that comes with the Unix version of a program called "Le Dico," which is available as:

```
        file://cipcinsa.insa-lyon.fr/apps/pub/france/ledico_u.zip
```

This version was already in Latin ISO. Based on my poor knowledge of French, Le Dico seems to do some dictionary maintenence and pattern matching. I don't know whether it does anything else.

```
        Le Dico came with a file called "copying.doc". I asked someone  who
knows French to tell me what it said, and then  I  looked  up  words  in  a
French-English  dictionary  to  bolster  my  understanding  of  it.  As   I
understand it, I am free to use  the  dictionary  as  I  wish.  If  I  were
modifying the source code to Le Dico or to its utility  programs,  I  would
have to include the complete sources. But I am not doing  that.  AlphaSpell
may do some of the same things as Le Dico, but it is not based on it in any
way. All I'm doing is making Le Dico's lexicon available for AlphaSpell  to
use.

        For those who know French, (and why would you want this  dictionary
if you don't?), here is the French text on distributing Le Dico:

   +--------------------[ Distribution De ]----------------------+
   |                                                              |
   |    LL       EEEEEEE        DDDDDD   IIII   CCCCC    OOOOO    |
   |    LL       EE             DD   DD   II   CC   CC  OO   OO   |
   |    LL       EEEE           DD   DD   II   CC       OO   OO   |
   |    LL       EE             DD   DD   II   CC       OO   OO   |
   |    LL       EE             DD   DD   II   CC   CC  OO   OO   |
   |    LLLLLLL  EEEEEEE        DDDDDD   IIII   CCCCC    OOOOO    |
   |                                                              |
   +--------------------------------------------------------------+

         Le Dico n'est pas domaine public, il est "Freeware".

         Considerez  que  Le  Dico  et  les  fichiers  le  composant  sont
         distribues pratiquement avec un "CopyLeft" similaire à  celui  des
         programmes GNU, parceque c'est un exemple du genre. (Mais Le  Dico
         n'a rien a voir avec GNU bien entendu).

         En resume, vous etes libres de diffuser gratuitement Le Dico a qui
         vous voulez. Vous  etes  libres  d'utiliser  tout  ou  partie  des
         sources et  fichiers  le  composant  pour  toute  realisation,  le
         lexique est d'ailleurs fait pour cela !

Distribution:

         Vous devez  fournir  l'archive  originale  complete,  sans  jamais
         dissocier les fichiers la composant. La version d' origine est  la
         version  Unix,  faites  circuler  cette  version  universelle   de
         preference a l'adaptation DOS chaque fois que c'est possible.

         Si vous faites des modifications ou ameliorations, ou realisez des
         utilitaires a distribuer avec Le Dico, vous devez le signaler dans
         la documentation, et fournir les SOURCES COMPLETES  (et  portables
         dans la mesure du possible) de tout ce que vous implementez, ou du
         moins laisser ces sources disponibles  a  tous  gratuitement,  sur
         simple demande.

         Ceci est  indispensable  pour  le  developpement  de  ce  type  de
         programme 'ouvert' et gratuit, devant etre accessible a  toute  la
         communaute.
```

1.95 German Dictionaries

```
FILE: German.lha
SHORT: German dictionary for AlphaSpell V
Type: text/edit
Uploader: fdnh@troi.cc.rochester.edu
Author: Fergus Duniho and Unknown
Version: 1.0

        Dies ist ein deutsches AlphaSpell Wörterbuch.

        This is a German AlphaSpell lexicon based on:

        ftp://sable.ox.ac.uk/pub/wordlists/german/germanl.Z

        I don’t know who originally compiled  the  wordlist.  The  original
wordlist was entirely in 7-bit ASCII, using ‘"’ after  vowels  to  indicate
umlauts, and ‘sS’ to indicate ‘ß’. So  I  wrote  a  conversion  script  for
Rainer Koppler’s program Cvt, which translated it to ISO Latin. That script
is included in this package.

        I came across a larger German dictionary at the  same  site,  which
had German words written entirely in Roman characters. Umlauted vowels were
followed by an e, and ss replaced ß. I’ve translated  the  umlauts,  but  I
don’t yet know how to decide which s-pairs should be converted into ß. Once
I’ve learned how, I’ll finish translating it and upload it.
```

1.96 Italian Dictionaries

```
        If you really need an Italian  dictionary,  you  can  download  and
adapt:

        ftp://sable.ox.ac.uk/pub/wordlists/italian/words.italian.Z

        But you should be warned that this wordlist is missing accents.  It
is for this reason that I have chosen not to base an AlphaSpell  dictionary
on it.
```

1.97 Latin Dictionaries

```
FILE: Latin.lha
SHORT: Latin dictionary for AlphaSpell V
Type: text/edit
Uploader: fdnh@troi.cc.rochester.edu
Author: Fergus Duniho and Unknown
Version: 1.0

        This is a Latin AlphaSpell dictionary based on:

        ftp://sable.ox.ac.uk/pub/wordlists/latin/wordlist.aug.Z
```

1.98 Norwegian Dictionaries

```
FILE: Norwegian.lha
SHORT: Norwegian dictionary for AlphaSpell V
Type: text/edit
Uploader: fdnh@troi.cc.rochester.edu
Author: Fergus Duniho and Unknown
Version: 1.0

        This is a Norwegian AlphaSpell dictionary based on:

        ftp://sable.ox.ac.uk/pub/wordlists/norwegian/words.norwegian.Z

        The README file  in  the  wordlists  directory  seems  to  indicate
that this wordlist was compiled by  Anders  Ellefsrud  <anders@ifi.uio.no>.
The original wordlist was entirely in 7-bit ASCII, using what I took to  be
the Norway character set.  So  I  wrote  a  conversion  script  for  Rainer
Koppler's program Cvt, which translates  ASCII  characters  in  the  Norway
character set to their 8-bit equivalents  in  ISO  Latin.  That  script  is
included in this package.
```

1.99 Spanish Dictionaries

```
FILE: Espanol.lha
SHORT: Spanish dictionary for AlphaSpell V
Type: text/edit
Uploader: fdnh@troi.cc.rochester.edu
Author: Joan Sola
Version: 1.0

        This dictionary was provided by Joan  Sola,  who  in  providing  it
became the first registered user of AlphaSpell. On  this  dictionary,  Joan
Sola writes, "I  work  as  a  translator  (English  ->  Spanish)  and  this
dictionary is the result of some massive extraction of  words  from  texts.
Spelling  is  good,  since  I  spell  checked  this  dic  text  in   a   PC
wordprocessor."
```

1.100 Swedish Dictionaries

```
FILE: Swedish.lha
SHORT: Swedish dictionary for AlphaSpell V
Type: text/edit
Uploader: fdnh@troi.cc.rochester.edu
Author: Fergus Duniho and Unknown
Version: 1.0

        This is a Swedish AlphaSpell dictionary based on:

        ftp://sable.ox.ac.uk/pub/wordlists/swedish/words.swedish.Z

        I don't know who originally compiled  the  wordlist.  The  original
wordlist was entirely in 7-bit ASCII, using what I took to be  the  Swedish
```

character set. So I wrote a conversion script for Rainer Koppler's program Cvt, which translates ASCII characters in the Swedish character set to their 8-bit equivalents in ISO Latin. That script is included in this package.

1.101 Icelandic Dictionaries

```
FILE: Icelandic.lha
SHORT: Icelandic dictionary for AlphaSpell V
Type: text/edit
Uploader: fdnh@troi.cc.rochester.edu
Author: Jorgen Pind
Version: 1.0
```

This dictionary was made from an Icelandic word list provided to me by Arni Freyr Jonsson in return for AlphaSpell's keyfile. The wordlist was made by Jorgen Pind for a program called stafs, which he made while working at "Orðabók Háskólans," The University of Iceland Dictionary. Jorgen Pind writes in Icelandic:

Það er guðvelkomið, orðalistinn er í public domain eða því sem næst. Sjálfur er ég höfundar listans og útbjó hann þegar ég starfaði á Orðabók Háskólans. Geta mætti þessa ef þið látið upplýsingar fylgja orðaskránni.

Kveðja,

Jörgen

```
------------------
Jorgen Pind                   Tel. +354-525-4086  Fax. +354-552-6806
Department of Psychology
University of Iceland         Internet: jorgen@rhi.hi.is
Oddi, 101 Reykjavik, Iceland
```

As translated into English by Arni Freyr Jonsson, this reads:

Yes, please do use my wordlist. It is practically public domain. I am the author of the wordlist myself and made it while working at The University of Iceland Dictionary. Please include this information with the distribution of your program.

1.102 Making a dictionary

There are basically two steps to creating an AlphaSpell dictionary:

(1) Acquiring a wordlist
(2) Converting a wordlist

1.103 Acquiring a wordlist

There are basically three ways to acquire a wordlist:

```
        Write one from scratch
        Generate one from word frequencies
        Find one by someone else
```

1.104 Writing a wordlist from scratch

The most effective way to do this is to type in words from a paper bound dictionary. This is a good way to create an accurate dictionary, but you had better be a fast typist to make the effort worthwhile.

1.105 Generating a wordlist from word frequencies

This is something AlphaSpell is designed for. It allows you to tabulate the word frequencies from countless reams of literature for creating a wordlist of commonly used words. There are many places on the Internet to find works of literature free for downloading. If you take this approach to creating a dictionary, here are some things to bear in mind. All the files you cull word frequencies from should be in the same language, and they should all use the same character set, preferably ISO Latin-1. If special characters are expressed with HTML codes or some other sort of coding, you should convert the files to a genuine character set before proceeding further. HTML codes and the like would screw things up, giving you misspelled words.

Suppose you download a bunch of literature from the Internet and put it all in a directory called lit. You could then type:

```
        AlphaSpell -o freqlist -Q lit/*
```

This would create a list of the word frequencies in the files. If you downloaded as many files as you had disk space for and want to add more, you can now delete the files, download more, and create a second list of frequencies. Using the -M command, you can merge the lists into one master list, then delete stuff and download more, repeating this cycle until you have a large enough list of word frequencies. You can merge frequency lists together like so:

```
        AlphaSpell -o freqlist -M freqlist flist2
```

If you use the -M command, you should randomize the lines in the files first, so that AlphaSpell will read them much faster.

Once you have a large frequency list, you can use the -W option to create a list of common words like so:

```
        AlphaSpell -o common -W freqlist -n 4000
```

1.106 Finding an already available wordlist

There are many wordlists available in:

ftp://sable.ox.ac.uk/pub/wordlists

If you're a college student there may be a wordlist available on the machine you read email on. But before you try to distribute it as an AlphaSpell dictionary, you should make sure there are no copyright restrictions on it that would prevent you from doing so.

You may find various dictionaries and wordlists on the Internet by entering "spell" or "dictionary" into a search engine such as SHASE or archie.

1.107 Converting a wordlist

Once you have a wordlist, you need to convert it into an AlphaSpell dictionary. Here is a guided example on how to do this. Suppose you come across a dictionary for Spangalese while exploring the web sites in Spanga.

The first thing you should do is make sure that it is in a character set recognized by AlphaSpell. AlphaSpell is designed to use and recognize the ISO Latin-1 character set. So make sure the wordlist uses this character set. If it doesn't convert it.

As it turns out, the Spangalese wordlist you found uses the IBM character set. To convert it, you may use cvt like so:

```
cvt Spangalese DSC DOSToAmi.cvt
```

Since the original DOSToAmi.cvt was missing some conversions, I've included an improved version with AlphaSpell.

Next, do a case sensitive sort on it. I use FSort and would type:

```
fsort Spangalese Spangalese case
```

Next, put all words with uppercase letters into Spangalese.mix by typing:

```
AlphaSpell -o Spangalese.mix -c -w *[A-ZÀ-Þ]* P Spangalese
```

The wildcard pattern includes the set of all capitalized letters. In ISO Latin-1, these are all the letters from A to Z and all the letters from À (capital A with a grave accent) to Þ (capital thorn). On the Amiga, you enter À by typing alt-g followed by A, and you enter Þ by typing alt-T.

Put the rest of the words into Spangalese.low by typing:

```
AlphaSpell -o Spangalese.low -c D Spangalese Spangalese.mix
```

After these steps, you should have a lowercase Spangalese dictionary, Spangalese.low, and a mixed case Spangalese dictionary, Spangalese.mix.

1.108 Index