Google

Code

Archive

babiloo - StarDict_format.wiki

Export to GitHub

Format for StarDict dictionary files

Extracted from the 3.0.0 source code

StarDict homepage: http://stardict.sourceforge.net

StarDict on-line dictionary: http://www.stardict.org

Number and Byte-order Conventions

``` When you record the numbers that identify sizes, offsets, etc., you should use 32-bits numbers, such as you might represent with a glong.

In order to make StarDict work on different platforms, these numbers must be in network byte order. You can ensure the correct byte order by using the g_htonl() function when creating dictionary files. Conversely, you should use g_ntohl() when reading dictionary files.

Strings should be encoded in UTF-8. ```

## Files

``` Every dictionary consists of these files: (1). somedict.ifo (2). somedict.idx or somedict.idx.gz (3). somedict.dict or somedict.dict.dz (4). somedict.syn (optional)

You can use gzip -9 to compress the .idx file. If the .idx file are not compressed, the loading can be fast and save memory when using, compress it will make the .idx file load into memory and make the quering become faster when using.

You can use dictzip to compress the .dict file. "dictzip" uses the same compression algorithm and file format as does gzip, but provides a table that can be used to randomly access compressed blocks in the file. The use of 50-64kB blocks for compression typically degrades compression by less than 10%, while maintaining acceptable random access capabilities for all data in the file. As an added benefit, files compressed with dictzip can be decompressed with gunzip. For more information about dictzip, refer to DICT project, please see: http://www.dict.org

When you create a dictionary, you should use .idx and .dict.dz in normal case.

Stardict will search for the .ifo file, then open the .idx or .idx.gz file and the .dict.dz or .dict file which is in the same directory and has the same base name.

~~~

## The ".ifo" file's format.

``` The .ifo file has the following format:

StarDict's dict ifo file version=2.4.2 [options]

Note that the current "version" string must be "2.4.2" or "3.0.0". If it's not, then StarDict will refuse to read the file. If version is "3.0.0", StarDict will parse the "idxoffsetbits" option.

## [options]

In the example above, [options] expands to any of the following lines specifying information about the dictionary. Each option is a keyword followed by an equal sign, then the value of that option, then a newline. The options may be appear in any order.

Note that the dictionary must have at least a bookname, a wordcount and a idxfilesize, or the load will fail. All other information is optional. All strings should be encoded in UTF-8.

Available options:

bookname= // required wordcount= // required synwordcount= // required if ".syn" file exists. idxfilesize= // required idxoffsetbits= // New in 3.0.0 author= email= website= description= // You can use
for new line. date= sametypesequence= // very important.

wordcount is the count of word entries in .idx file, it must be right.

idxfilesize is the size(in bytes) of the .idx file, even the .idx is compressed to a .idx.gz file, this entry must record the original .idx file's size, and it must be right too. The .gz file don't contain its original size information, but knowing the original size can speed up the extraction to memory, as
~~~

you don't need to call realloc() for many times.

idxoffsetbits can be 64 or 32. If "idxoffsetbits=64", the offset field of the .idx file will be 64 bits.

The "sametypesequence" option is described in further detail below.

---

sametypesequence

You should first familiarize yourself with the .dict file format described in the next section so that you can understand what effect this option has on the .dict file.

If the sametypesequence option is set, it tells StarDict that each word's data in the .dict file will have the same sequence of datatypes. In this case, we expect a .dict file that's been optimized in two ways: the type identifiers should be omitted, and the size marker for the last data entry of each word should be omitted.

Let's consider some concrete examples of the sametypesequence option.

Suppose that a dictionary records many .wav files, and so sets: sametypesequence=W In this case, each word's entry in the .dict file consists solely of a wav file. In the .dict file, you would leave out the 'W' character before each entry, and you would also omit the 32-bits integer at the front of each .wav entry that would normally give the entry's length. You can do this since the length is known from the information in the idx file.

As another example, suppose a dictionary contains phonetic information and a meaning for each word. The sametypesequence option for this dictionary would be: sametypesequence=tm Once again, you can omit the 't' and 'm' characters before each data entry in the .dict file. In addition, you should omit the terminating '\0' for the 'm' entry for each word in the .dict file, as the length of the meaning string can be inferred from the length of the phonetic string (still indicated by a terminating '\0') and the length of the entire word entry (listed in the .idx file).

So for cases where the last data entry for each word normally requires a terminating '\0' character, you should omit this character in the dict file. And for cases where the last data entry for each word normally requires an initial 32-bits number giving the length of the field (such as WAV and PNG entries), you must omit this number in the dictionary.

Every dictionary should try to use the sametypesequence feature to save disk space.

---

~~~

## The ".idx" file's format

``` The .idx file is just a word list.
~~~

The word list is a sorted list of word entries.

Each entry in the word list contains three fields, one after the other: word_str; // a utf-8 string terminated by '\0'. word_data_offset; // word data's offset in .dict file word_data_size; // word data's total size in .dict file

word_str gives the string representing this word. It's the string that is "looked up" by the StarDict.

Two or more entries may have the same "word_str" with different word_data_offset and word_data_size. This may be useful for some dictionaries. But this feature is only well supported by StarDict-2.4.8 and newer.

The length of "word_str" should be less than 256. In other words, (strlen(word) < 256).

If the version is "3.0.0" and "idxoffsetbits=64", word_data_offset will be 64-bits unsigned number in network byte order. Otherwise it will be 32-bits. word_data_size should be 32-bits unsigned number in network byte order.

It is possible the different word_str have the same word_data_offset and word_data_size, so multiple word index point to the same definition. But this is not recommended, for mutiple words have the same definition, you may create a ".syn" file for them, see section 4 below.

The word list must be sorted by calling stardict_strcmp() on the "word_str" fields. If the word list order is wrong, StarDict will fail to function correctly!

============ gint stardict_strcmp(const gchar *s1, const gchar *s2) { gint a; a = g_ascii_strcasecmp(s1, s2); if (a == 0) return strcmp(s1, s2); else return a;

# }

g_ascii_strcasecmp() is a glib function: Unlike the BSD strcasecmp() function, this only recognizes standard ASCII letters and ignores the locale, treating all non-ASCII characters as if they are not letters.

stardict_strcmp() works fine with English characters, but the other locale characters' sorting is not so good, in this case, you can enable the collation feature, see section 6.

~~~

## The ",syn" file's format.

``` This file is optional, and you should notice tree dictionary needn't this file. Only StarDict-2.4.8 and newer support this file.
~~~

The .syn file contains information for synonyms, that means, when you input a synonym, StarDict will search another word that related to it.

The format is simple. Each item contain one string and a number. synonym_word; // a utf-8 string terminated by '\0'. original_word_index; // original word's index in .idx file. Then other items without separation. When you input synonym_word, StarDict will search original_word;

The length of "synonym_word" should be less than 256. In other words, (strlen(word) < 256). original_word_index is a 32-bits unsigned number in network byte order. Two or more items may have the same "synonym_word" with different original_word_index. The items must be sorted by stardict_strcmp() with synonym_word.

## The offset cache file's format

``` StarDict-2.4.8 start to support cache files, this feature can speed up loading and save memory as mmap() the cache file. The cache file names are .idx.oft and .syn.oft, the format is: First a utf-8 string terminated by '\0', then many 32-bits numbers as the wordoffset index, this index is sparse, and "ENTR_PER_PAGE=32", they are not stored in network byte order.

# The string must begin with:

StarDict's oft file

# version=2.4.8

Then a line like this: url=/usr/share/stardict/dic/stardict-somedict-2.4.2/somedict.idx This line should have a ending '\n'.

StarDict will try to create the .oft file at the same directory of the .ifo file first, if failed, then try to create it at ~/.cache/stardict/, ~/.cache is get by g_get_user_cache_dir(). If two or more dictionaries have the same file name, StarDict will create somedict.idx.oft, somedict(2).idx.oft, somedict(3).idx.oft, etc. for them respectively, each with different "url=" in the beginning string.

```

The collation file's format

``` StarDict-2.4.8 start to support collation, that sort the word list by collate function. It will create collation file which names .idx.clt and .syn.clt, the format is a little like offset cache file: First a
```

utf-8 string terminated by '\0', then many 32-bits numbers as the index that sorted by the collate function, they are not stored in network byte order.

The string must begin with:

StarDict's clt file

version=2.4.8

Then two lines like this: url=/usr/share/stardict/dic/stardict-somedict-2.4.2/somedict.idx func=0 The second line should have a ending '\n' too.

StarDict support these collate functions currently: typedef enum { UTF8_GENERAL_CI = 0, UTF8_UNICODE_CI, UTF8_BIN, UTF8_CZECH_CI, UTF8_DANISH_CI, UTF8_ESPERANTO_CI, UTF8_ESTONIAN_CI, UTF8_HUNGARIAN_CI, UTF8_ICELANDIC_CI, UTF8_LATVIAN_CI, UTF8_LITHUANIAN_CI, UTF8_PERSIAN_CI, UTF8_POLISH_CI, UTF8_ROMAN_CI, UTF8_ROMANIAN_CI, UTF8_SLOVAK_CI, UTF8_SLOVENIAN_CI, UTF8_SPANISH_CI, UTF8_SPANISH2_CI, UTF8_SWEDISH_CI, UTF8_TURKISH_CI, COLLATE_FUNC_NUMS } CollateFunctions; These UTF8_*_CI functions comes from MySQL in fact.

The file's locate path just like the .oft file.

Notice, for "somedict.idx.gz" file, the corresponding collation file is somedict.idx.clt, but not somedict.idx.gz.clt, the "url=" is somedict.idx, not somedict.idx.gz. So after you gzip the .idx file, StarDict needn't create the .clt file again.

~~~

The ".dict" file's format

``` The .dict file is a pure data sequence, as the offset and size of each word is recorded in the corresponding .idx file.

If the "sametypesequence" option is not used in the .ifo file, then

# the .dict file has fields in the following order:

word_1_data_1_type; // a single char identifying the data type word_1_data_1_data; // the data word_1_data_2_type; word_1_data_2_data; ...... // the number of data entries for each word is determined by // word_data_size in .idx file word_2_data_1_type; word_2_data_1_data;
~~~

## ......

It's important to note that each field in each word indicates its own length, as described below. The number of possible fields per word is also not fixed, and is determined by simply reading data until you've read word_data_size bytes for that word.

Suppose the "sametypesequence" option is used in the .idx file, and the option is set like this: sametypesequence=tm

# Then the .dict file will look like this:

word_1_data_1_data word_1_data_2_data word_2_data_1_data word_2_data_2_data

## ......

The first data entry for each word will have a terminating '\0', but the second entry will not have a terminating '\0'. The omissions of the type chars and of the last field's size information are the optimizations required by the "sametypesequence" option described above.

If "idxoffsetbits=64", the file size of the .dict file will be bigger than 4G. Because we often need to mmap this large file, and there is a 4G maximum virtual memory space limit in a process on the 32 bits computer, which will make we can get error, so "idxoffsetbits=64" dictionary can't be loaded in 32 bits machine in fact, StarDict will simply print a warning in this case when loading. 64-bits computers should haven't this limit.

## Type identifiers

Here are the single-character type identifiers that may be used with the "sametypesequence" option in the .idx file, or may appear in the dict file itself if the "sametypesequence" option is not used.

Lower-case characters signify that a field's size is determined by a terminating '\0', while upper-case characters indicate that the data begins with a network byte-ordered guint32 that gives the length of the following data's size(NOT the whole size which is 4 bytes bigger).

'm' Word's pure text meaning. The data should be a utf-8 string ending with '\0'.

'l' Word's pure text meaning. The data is NOT a utf-8 string, but is instead a string in locale encoding, ending with '\0'. Sometimes using this type will save disk space, but its use is discouraged.

'g' A utf-8 string which is marked up with the Pango text markup language. For more information about this markup language, See the "Pango Reference Manual." You might have it installed locally at: file:///usr/share/gtk-doc/html/pango/PangoMarkupFormat.html

't' English phonetic string. The data should be a utf-8 string ending with '\0'.

Here are some utf-8 phonetic characters: θʃŋʧðʒæıʌʊɒɛəɑɜɔˌˈːˑṃṇḷ æɑɒʌəєŋvθðʃʒɚːɡˏˊˋ

'x' A utf-8 string which is marked up with the xdxf language. See http://xdxf.sourceforge.net StarDict have these extention: can have "type" attribute, it can be "image", "sound", "video" and "attach". can have "k" attribute.

'y' Chinese YinBiao or Japanese KANA. The data should be a utf-8 string ending with '\0'.

'k' KingSoft PowerWord's data. The data is a utf-8 string ending with '\0'. It is in XML format.

'w' MediaWiki markup language. See http://meta.wikimedia.org /wiki/Help:Editing#The_wiki_markup

'h' Html codes.

'r' Resource file list. The content can be: img:pic/example.jpg // Image file snd:apple.wav // Sound file vdo:film.avi // Video file att:file.bin // Attachment file More than one line is supported as a list of available files. StarDict will find the files in the Resource Storage. The image will be shown, the sound file will have a play button. You can "save as" the attachment file and so on.

'W' wav file. The data begins with a network byte-ordered guint32 to identify the wav file's size, immediately followed by the file's content.

'P' Picture file. The data begins with a network byte-ordered guint32 to identify the picture file's size, immediately followed by the file's content.

'X' this type identifier is reserved for experimental extensions.

~~~

## Resource Storage

``` Resource Storage store the external file in 'r' resource file list, the image in html code, the image, media and other files in wiki tag. It have two forms: 1. Direct directory and files in the "res" sub-directory. 2. The res.rifo, res.ridx and res.rdic database. Direct files may have file name encoding problem, as Linux use UTF-8 and Windows use local encoding, so you'd better just use ASCII file name, or use databse to store UTF-8 file name. Databse may need to extract the file(such as .wav) file to a temporary file, so not so efficient compare to direct files. But database have the advantage of compressing. You can convert the res directory and the res database
~~~

from each other by the dir2resdatabse and resdatabase2dir tools. StarDict will try to load the storage database first, then try the direct files form.

The format of the res.rifo file: StarDict's storage ifo file version=3.0.0 filecount= // required. idxoffsetbits= // optional.

The format of the res.ridx file: filename; // A string end with '\0'. offset; // 32 or 64 bits unsigned number in network byte order. size; // 32 bits unsigned number in network byte order. filename can include a path too, such as "pic/example.png". filename is case sensitive, and there should have no two same filenames in all the entries. if "idxoffsetbits=64", then offset is 64 bits. These three items are repeated as each entry. The entries are sorted by the strcmp() function with the filename field. It is possible that different filenames have the same offset and size.

The format of the res.rdic file: It is just the join of each resource files. You can dictzip this file as res.rdic.dz

~~~

## Tree Dictionary

``` The tree dictionary support is used for information viewing, etc.

A tree dictionary contains three file: sometreedict.ifo, sometreedict.tdx.gz and sometreedict.dict.dz.

It is better to compress the .tdx file, as it is always load into memory.

The .ifo file has the following format:

StarDict's treedict ifo file version=2.4.2 [options]

Available options:

bookname= // required tdxfilesize= // required wordcount= author= email= website= description= date= sametypesequence=

wordcount is only used for info view in the dict manage dialog, so it is not important in tree dictionary.

## The .tdx file is just the word list.

The word list is a tree list of word entries.

Each entry in the word list contains four fields, one after the other: word_str; // a utf-8 string terminated by '\0'. word_data_offset; // word data's offset in .dict file word_data_size; // word
~~~

data's total size in .dict file. it can be 0. word_subentry_count; //how many sub word this entry has, 0 means none.

Subentry is immidiately followed by its parent entry. This make the order is just as when a tree list with all its nodes extended, then sort from top to bottom.

word_data_offset, word_data_size and word_subentry_count should be 32-bits unsigned numbers in network byte order.

The .dict file's format is the same as the normal dictionary.

```

More information

You can read "src/lib.cpp", "src/dictmanagedlg.cpp" and "src/tools/**.cpp" for more information.**

After you have build a dictionary, you can use "stardict_verify" to verify the dictionary files. You can find it at "src/tools/".

If you have any questions, email me. :)

Thanks to Will Robinson for cleaning up this file's English.

Hu Zheng http://forlinux.yeah.net 2007.4.24
```