# Friso 开发帮助文档

(注:该文档只适合 friso 1.2 及以上的版本)

## 一. 关于 *friso:*

Friso 是使用 c 语言开发的一款中文分词器，使用流行的 mmseg 算法实现。完全基于模块化设计和实现，可以很方便的植入到其他程序中，例如：MySQL，PHP 等。源码无需修改就能在各种平台下编译使用，加载完 20 万的词条，内存占用稳定为 14.5M。

官方首页: https://code.google.com/p/friso

下载地址: https://code.google.com/p/friso/downloads/list

Friso 最新版本功能说明：（可以略过）

1。目前最高版本：friso 1.6.0，同时支持对 **UTF-8/GBK** 编码的切分。

2。mmseg 四种过滤算法，分词准确率达到了 98.41%，请参考本算法的原作：http://technology.chtsai.org/mmseg/。

3。支持自定义词库。在 dict 文件夹下，可以随便添加/删除/更改词库和词库词条，并且对词库进行了分类。

4。简体/繁体/简体混合支持, 可以方便的针对简体，繁体或者简繁体切分。同时还可以以此实现简繁体的相互检索。

5。支持中英/英中混合词的识别(维护词库可以识别任何一种组合)。例如：卡拉 ok, 漂亮 mm, c 语言，IC 卡，哆啦 a 梦。

7。很好的英文支持，英文标点组合词识别, 例如 c++, c#, 电子邮件，网址，小数，百分数。

8。(**!New**)自定义保留标点：你可以自定义保留在切分结果中的标点，这样可以识别出一些复杂的组合，例如：c++, k&r，code.google.com。

9。(**!New**)复杂英文切分的二次切分：默认 Friso 会保留数字和字母的原组合，开启此功能，可以进行二次切分提高检索的命中率。例如：qq2013 会被切分成：qq/ 2013/ qq2013。

10。支持阿拉伯数字/小数基本单字单位的识别，例如 2012 年，1.75 米，5 吨，120 斤，38.6℃。

11。自动英文圆角/半角，大写/小写转换。

12。同义词匹配：自动中文/英文同义词追加. (需要在 friso.ini 中开启 friso.add_syn 选项)。

13。自动中英文停止词过滤。(需要在 friso.ini 中开启 friso.clr_stw 选项)。

14。多配置支持, 安全的应用于多进程/多线程环境。

15。提供 friso.ini 配置文件, 可以依据你的需求轻松打造适合于你的应用的分词。

升级的详细功能变化，请查看附件里面的 **CHANGES.md** 文件。

## 二. 安装 *friso:*

先到 friso 官方网站下载最新版本的 friso: friso-{version}-src-dict.zip, {version} 表示版本号, 下同.

解压 friso-{version}-src-dict.zip 到{path}, 设{path}为你的解压后 friso 的根目录, 下同:

**1. Linux, 在终端 cd 到{path}/src 目录后, 然后依次运行:**

make

sudo make install

**2. WinNT:**

(1). 使用 VS 编译得到 dll 和 lib 文件，具体可以参考 Friso 讨论：
http://www.oschina.net/question/853816_135216

(2). 直接使用{path}/lib/win32 下的 friso.dll (默认使用编译(推荐))

(3). 使用 cygwin 从源码编译, 安装好 cygwin 后, 删除原有的 Makefile, 更改 Makefile.cygwin 为 Makefile, 打开 cygwin 的终端, cd 到{path}/src 下:

运行: make

在{path}/src 下即可得到 friso.exe 和 friso.dll

## 三. 配置 *friso:*

Friso 要做的配置工作很简单: 打开{path}目录, 找到 friso.ini 配置文件, 使用文本编辑器打开即可.

找到 friso.lex_dir, 修改其值为词库目录绝对地址, 并且必须以"/"结尾. 即:

**friso.lex_dir = {path}/dict/GBK 或者 UTF-8/**

例如: (回想第二步)假设你的 friso 解压在/opt/friso 下，使用 UTF-8 编码, 则:

friso.lex_dir = /opt/friso/dict/UTF-8/

**friso.ini 配置文件:(可以不用理会)**

```
#Friso 使用的切分编码。(0 表示 UTF-8, 1 表示 GBK, Friso-1.6.0)
friso.charset = 0

#词库绝对路径(注意词库分为 UTF-8 和 GBK)
friso.lex_dir = /c/products/friso/dict/UTF-8/

#最大匹配长度
friso.max_len = 5
```

```
#是否开启中文姓名识别(目前还不支持)
friso.r_name = 1

#英中混合词中文词数
friso.mix_len = 2

#中文姓氏修饰词长度
friso.lna_len = 1

#是否追加同义词
friso.add_syn = 1

#是否保留不识别的词条 (1 保留, 0 直接过滤)
#@date 2013-06-13
friso.keep_urec = 0

#是否启用 sphinx 定制输出(1 开启, 0 关闭)
#@date 2013-10-25
friso.spx_out = 0

#是否过滤停止词
friso.clr_stw = 0

#开启复杂英文的二次切分 (Friso-1.6.0)
friso.en_sseg = 1

#二次切分 sub Token 最小长度 (Friso-1.6.0)
friso.st_minl = 2

#英文切分过程中默认保留的标点 (Friso-1.6.0)
friso.kpuncs = @%.#&+

#用于姓名识别中的阈值.
friso.nthreshold = 2000000
```

```
#切分模式(1-简易模式, 2-复杂模式)
friso.mode = 2
```

## 四. 运行测试程序:

**1. Linux,** 在终端直接运行:

friso -init {path}/friso.ini

**2. WinNT:** ( 注意编码要设置为 **GBK** )

(1). cygwin 编译的, 在 cygwin 终端直接运行:

./friso -init {path}/friso.ini

(2). 没有 cygwin, 拷贝{path}/lib/win32 下的 friso.dll 到环境路径 path 下后, 载入 friso.dll, 然后使用 vc 或者 vs 编译运行{path}/src/tst-friso.c.

运行成功后你会看到如下的操作界面:

```
friso initialized in 0.160000sec
+-------------------------------------------------------------------------------+
| friso - a chinese word segmentation writen by c.          |
| bug report email - chenxin619315@gmail.com.               |
| or: visit http://code.google.com/p/friso.                 |
|     java edition for http://code.google.com/p/jcseg       |
| type 'quit' to exit the program.                          |
+-------------------------------------------------------------------------------+
friso>>
```

在提示 friso>>后输入你要分词的内容按 Enter 即可.

例如:

```
friso>> 研究生命起源, i love c++.
分词结果:
研究/ 琢磨/ 研讨/ 钻研/ 生命/ 起源/ ,/ i/ love/ c++/ ./
Done, cost < 0.000000sec
```

注意 1: 提示界面的第一行有个: friso initialized in 0.160000sec, 如果是 friso initialized in 0.000000sec, 那就一定没有配置好, 也就是 friso 没有正确的加载词库, 确保 friso.ini 中的 friso.lex_dir 指向正确的词库目录。

注意 2: 如果直接使用 WinNT 下的 cmd 运行测试程序，注意要在 friso.ini 中设置编码为 GBK，也就是 friso.charset = 1，同时设置词库路径为 GBK 词库路径。

## 五. 二次开发*(friso api):*

要使用 friso 来进行分词，你需要两个对象: **friso_t**(friso_entry)对象和一个 **friso_task_t**(friso_task_entry)对象. 两者都在 friso.h 头文件中定义的:

(可以先看下面的"一个完整的例子")

**1. friso_t 对象:**

定义:

```
//friso-1.5.0 版本以上版本:
typedef struct {
    ushort_t max_len;     //the max match length (4 - 7).
    ushort_t r_name;      //1 for open cn recognition 0 for close it.
    ushort_t mix_len;     //the max length for the CJK words in a mix string.
    ushort_t lna_len;     //the max length for the chinese last name adron.
    ushort_t add_syn;     //append synonyms tokenizer words.
    ushort_t clr_stw;     //clear the stopwords.
    uint_t nthreshold;    //the threshold for a char to make up a cn name.
    friso_mode_t mode;    //Complex mode or simple mode
    friso_dic_t dic;      //friso dictionary
} friso_entry;
typedef friso_entry * friso_t;

//friso-1.5.0 以及一下版本:
/* friso entry.*/
typedef struct {
    friso_dic_t dic;                //friso dictionary
    friso_charset_t charset;        //friso charset.
} friso_entry;
typedef friso_entry * friso_t;

//其中的重点就是 dic, 也就是 friso 的词库.

/*新加入的 friso_config_entry 配置实例*/
```

```
    typedef struct {
        ushort_t max_len;     //the max match length (4 - 7).
        ushort_t r_name;      //1 开启中文人名识别，0 关闭.
         ushort_t mix_len;    //the max length for CJK word in a mix string.
        ushort_t lna_len;     //the max length for the chinese last name adron.
        ushort_t add_syn;     //append synonyms tokenizer words.
        ushort_t clr_stw;     //clear the stopwords.
        ushort_t keep_urec; //keep the unrecongnized words.
        ushort_t spx_out;     //use sphinx output customize.
        uint_t nthreshold;    //the threshold value for char to make up a chinese
name.
        ushort_t en_sseg;     //start the secondary segmentation.(Friso-1.6.0)
        ushort_t st_minl;     //min length of the sub token.(Friso-1.6.0)
        friso_mode_t mode;  //Complex mode or simple mode
        char kpuncs[_FRISO_KEEP_PUNC_LEN]; //keep punctuations buffer.
(Friso-1.6.0)
    } friso_config_entry;
    typedef friso_config_entry * friso_config_t;
```

**创建：**

friso 内部提供了 api 来创建并且初始化 friso_entry 的函数:

(1). 直接创建从 friso.ini 中初始化的 friso_entry 并且依据 friso.ini 中的配置自动创建和加载词库:

```
    //__path__为 friso.ini 文件的有效地址. (friso-1.5.0 以前的版本)
    friso_t friso = friso_new_from_ifile(__path__);
```

(2). 单独创建并且设置:

```
    //创建 friso_t
    friso_t friso = friso_new();

    //创建词库 dic(并没有加载词库)
    friso->dic = friso_dic_new();
```

```
//给词库加载词条
//1.从给定的 friso.lex.ini 中加载词库({path}/dict/下有个 friso.lex.ini)
//friso_dic_load_from_ifile(friso, fstring, uint_t)
//friso:      当前工作的 friso 实例
//fstring:    friso.lex.ini 文件的有效地址
//uint_t:     词条长度最大长度限制(字节数, 例如最大词长 5 字, 则为 3*5 个字节)
//2. 也可以逐个类别的加载词库:
//friso_dic_load(friso_t, friso_lex_t, fstring, uint)
friso_dic_load_from_ifile(friso, __path__, 15);


//###friso-1.5.0 及以后的版本:
friso_dic_load_from_ifile(friso, config, __path__, 15);

//friso_t 的其他 api
//1.设置切分模式(简易和复杂)
//__FRISO_SIMPLE_MODE__和__FRISO_COMPLEX_MODE__
friso_set_mode(friso, __FRISO_COMPLEX_MODE__);
```

释放:

friso_t 实例用完后需要使用如下 api 来释放:

```
//释放 friso_t 实例
friso_free(friso);
```

**2. friso_task_t 对象:**

定义:

```
typedef struct {
    fstring text;           //text to tokenize
    uint_t idx;             //start offset index.
    uint_t length;          //length of the text.
    uint_t bytes;           //latest word bytes in C.
    uint_t unicode;         //latest word unicode number.
```

```
    friso_link_t pool;      //task pool.
    string_buffer_t sbuf; //string buffer. (Friso-1.6.0)
    friso_hits_t hits;      //token result hits.
    char buffer[7];         //word buffer. (1-6 bytes for an utf-8 word in C).
} friso_task_entry;
typedef friso_task_entry * friso_task_t;
```

text 指向需要被切分的 utf-8 编码的字符串.
idx 表示下一个切分的开始索引.
length 表示字符串的长度(字节).
hits 表示一个切分结果.
pool 切分结果缓冲池(一个链表).
其他的是一些为方便中间过程切分的辅助变量.

再来看下 hits(friso_hits_t)的结构:

```
typedef struct {
    uchar_t type;      //type of the word. (item of friso_lex_t) (Friso-1.6.0)
    uchar_t length;   //length of the token. (Friso-1.6.0)
    uchar_t rlen;      //the real length of the token.(Friso-1.6.0)
    char pos;          //part of speech.  (Friso-1.6.0)
    int offset;
    char word[__HITS_WORD_LENGTH__];
} friso_hits_entry;
typedef friso_hits_entry * friso_hits_t;
```

friso_hits_t 是用来保存一个切分结果.
type 是词条类别。
length 是词条长度。（Friso 内部优化之后）
rlen 词条真实长度。(Friso 内部优化前)
pos 词条词性。(待实现。)
offset 是这个切分到的词在整个字符串中的偏移量。
word 即为这个词。

**创建：**
同样的, friso 内部提供了 api 来创建 friso_task_t.

```
//创建一个分词任务实例
friso_task_t task = friso_new_task();

//给分词任务设置分词的内容.
fstring text = “研究生命起源”;
friso_set_text( task, text );
```

**释放：**

同样的, 用完的 friso_task_t 需要调用下面的 api 来释放:

```
//释放 friso_task_t 实例
friso_free_task( task );
```

**3.** 看一个完整的例子：

接下我们使用 friso_t 和 friso_task_t 来写一个完整的列子

详细查看源码中 **tst-friso.c** 完整的样板：

```
//friso-1.5.0 以前的版本：
//1.创建一个 friso_t 实例
friso_t friso = friso_new_from_ifile(“/c/friso/friso.ini”);
//2.创建一个 friso_task_t 实例:
friso_task_t task = friso_new_task();

//3.给分词任务 task 设置分词内容
fstring text = “这里是要被分词的字符串”;
friso_set_text( task, text );

//4.获取切分结果
//friso_next 获取下一个切分结果
//得到的切分结果存放在 task->hits 中.
//通过 task->hits->word 的到切分的词条.
//通过 task->hits->offset 得到对应词条在原文中的偏移位置.
```

```
while ( ( friso_next( friso, friso->mode, task ) ) != NULL ) {
    //printf("%s[%d]/ ", task->hits->word, task->hits->offset );
    printf("%s/ ", task->hits->word );
}

//5.释放
friso_free_task( task );
friso_free(friso);

//friso-1.5.0 以及以后的版本：
friso_t friso = friso_new();
friso_config_t config = friso_new_config();
//从指定的 friso.ini 文件中初始化 friso 和 config.
friso_init_from_ifile(friso, config, _ifile);

//创建分词任务&&设置分词内容
friso_task_t task = friso_new_task();
fstring text = “这里是要被分词的字符串”;
friso_set_text( task, text );

//.获取切分结果
//friso_next 获取下一个切分结果
//得到的切分结果存放在 task->hits 中.
//通过 task->hits->word 的到切分的词条.
//通过 task->hits->offset 得到对应词条在原文中的偏移位置.
while ( ( friso_next( friso, config, task ) ) != NULL ) {
    //查看 friso_hits_t 可以获取更多信息。
    //printf("%s[%d]/ ", task->hits->word, task->hits->offset );
    printf("%s/ ", task->hits->word );
}

//释放资源...
friso_free_task( task );
friso_free_config(config);
friso_free(friso);
```

注意: 在单线程环境下可以反复的利用创建的 friso_t, friso_config_t(1.5.0 以及以上)和 friso_task_t. 切分不同的内容的时候调用 friso_set_text(friso_task_t, fstring)来重置 friso_task_t 的切分内容即可. {path}/src/tst-friso.c 是一个完整的例子.

而, 在多线程环境下: 不同线程共用一个 friso_t, 每个线程都创建一个 friso_task_t. 具体例子, 可以查看基于 friso 的 php 中文分词扩展- robbe (http://code.google.com/p/robbe)

## 六. 词库管理:

Friso 内部对词库进行了分类, 在管理词库前你需要先了解这些分类:

**friso 词库类别:**

```
typedef enum {
    __LEX_CJK_WORDS__ = 0,          //普通 CJK 词库
    __LEX_CJK_UNITS__ = 1,          //CJK 单位词库
    __LEX_ECM_WORDS__ = 2,          //英中混合词(例如: b 超)
    __LEX_CEM_WORDS__ = 3,          //中英混合词(例如: 卡拉 ok).
    __LEX_CN_LNAME__ = 4,           //中文姓氏
    __LEX_CN_SNAME__ = 5,           //中文单姓名词库
    __LEX_CN_DNAME1__ = 6,          //中文双姓名首字词库
    __LEX_CN_DNAME2__ = 7,          //中文双姓名尾字词库
    __LEX_CN_LNA__ = 8,             //中文姓氏修饰词词库
    __LEX_STOPWORDS__ = 9,          //停止词词库
    __LEX_ENPUN_WORDS__ = 10,       //英文和标点混合词库(例如: c++)
    __LEX_OTHER_WORDS__ = 15,       //无用
    __LEX_NCSYN_WORDS__ = 16        //无用
} friso_lex_t;
```

再来看看 friso.lex.ini 配置文件:

```
#main lexion
__LEX_CJK_WORDS__ :[
    lex-main.lex;
    lex-admin.lex;
    lex-chars.lex;
    lex-cn-mz.lex;
```

```
    lex-cn-place.lex;
    lex-company.lex;
    lex-festival.lex;
    lex-flname.lex;
    lex-food.lex;
    lex-lang.lex;
    lex-nation.lex;
    lex-net.lex;
    lex-org.lex;
    #add more here
]
#single chinese unit lexicon
__LEX_CJK_UNITS__ :[
    lex-units.lex;
]
#chinese and english mixed word lexicon like "b 超".
__LEX_ECM_WORDS__:[
    lex-ecmix.lex;
]
#english and chinese mixed word lexicon like "卡拉 ok".
__LEX_CEM_WORDS__:[
    lex-cemix.lex;
]
#chinese last name lexicon.
__LEX_CN_LNAME__:[
    lex-lname.lex;
]
#single name words lexicon.
__LEX_CN_SNAME__:[
    lex-sname.lex;
]
#first word of a double chinese name.
__LEX_CN_DNAME1__:[
    lex-dname-1.lex;
]
#second word of a double chinese name.
```

```
__LEX_CN_DNAME2__:[
    lex-dname-2.lex;
]
#chinese last name decorate word.
__LEX_CN_LNA__:[
    lex-lna.lex;
]
#stopwords lexicon
__LEX_STOPWORDS__:[
    lex-stopwords.lex;
]
#english and punctuation mixed words lexicon.
__LEX_ENPUN_WORDS__:[
    lex-en-pun.lex;
]
```

格式如下:

```
词库类别关键字: [
    词库文件;
]
```

上面的 10 个词库类被关键字分别对应于 friso 的 10 个词库类别, []中的内容就是该类别的词库文件, 一个类别可以有多个词库文件. 类别是系统定义的, 不能随便添加.

**1.** 加入新词库文件**:**

首先确认你要加入的词库文件的类别.

例如: 我想添加一个词库文件专门用来存储植物的名字, 在{path}/dict/下新建 lex-pname.lex, 然后按照一个词条一行的规则加入词条到 lex-pname.lex 来完善该词库.

接下来你还有一个重要的步骤就是将该词库归类到 friso.lex.ini 中去, 通常的词库都是 CJK 词库, 也就是将 lex-pname.lex 作为一行加入到:

```
__LEX_CJK_WORDS__:[
    lex-main.lex;
    lex-admin.lex;
    lex-chars.lex;
    lex-cn-mz.lex;
    lex-cn-place.lex;
    lex-company.lex;
```

```
        lex-festival.lex;
        lex-flname.lex;
        lex-food.lex;
        lex-lang.lex;
        lex-nation.lex;
        lex-net.lex;
        lex-org.lex;
        lex-pname.lex;
#add more here
    ]
```

新词库文件的加入工作就 bingo 了.

**2.** 在给定词库文件中加入新词条**:**

这个工作做起来太简单了, 找到对应的词库文件, 使用文本编辑器打开, 将要加入的词条按照下面的格式作为一行加入即可. (Tip: 加入前先确认下相同的词条不存在, 重复存在也没关系, 只不过会浪费磁盘空间并且会影响词库的加载时间).

Friso 词库词条格式:

```
词条/同义词集合
```

同义词没有使用 null 代替, 多个同义词使用英文逗号隔开.

例如: 研究

研究/琢磨,研讨,钻研

3. 繁体/简繁体混合支持: (friso-1.5.0 以及以上版本):

在 friso 官网下载最新的全部词库, simplified 是简体词库, traditional 是繁体词库, mixed 是简繁体混合词库, 依据你的需求选择对应的词库就可以了.

## 七. 联系作者**:**

作者信息: 陈鑫 – 网名: 狮子的魂
电子邮件: chenxin619315@gmail.com

## 八. 更多开源软件**:**

1. java 开源中文分词分词器 – jcseg
   http://code.google.com/p/jcseg

2. 基于 friso 实现的开源 php 中文分词扩展 – robbe
   http://code.google.com/p/robbe
3. 开源跨平台多媒体教学软件 – jteach
   http://code.google.com/p/jteach