### Rime中州韻输入法配置手册

**前言：**

工欲善其事，必先利其器！Rime中州韻输入法准确地讲是一个输入法框架，它可自由地自定义输入的方式及内容，即我们理解的输入法 在rime这里只是一个配置方案，rime框架下可有多个输入法方案。Rime最初只有linux版本的，后来有了windows及mac版本，在不同系统下，其软件名称不一样:

windows: weasel 小狼毫

linux: ibus-rime 中州韻

macOS: squirrel 鼠鬚管

**官网：** https://rime.im/

**开源地址：** https://github.com/rime/

**Rime作者：** 佛振 （神一般的输入法，伟大的发明，向作者致敬）

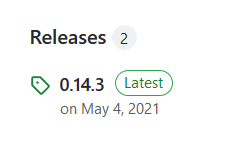
本文档重点讲解rime的windows版本（weasel 小狼毫）的使用及自定义输入法方案

作者：李茂福

日期：2022年6月4日

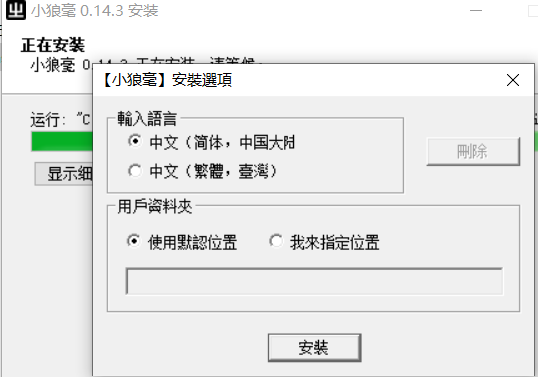
**章0、下载及安装**

weasel小狼毫软件下载地址： https://github.com/rime/weasel

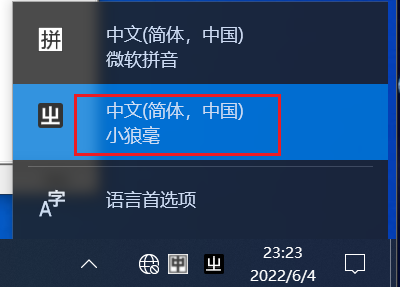


https://github.com/rime/weasel/releases/download/0.14.3/weasel-0.14.3.0-installer.exe

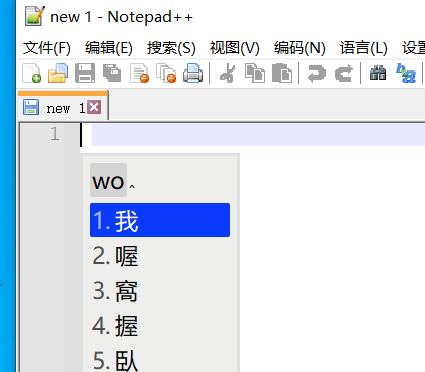
下载后，双击安装，默认选项即可



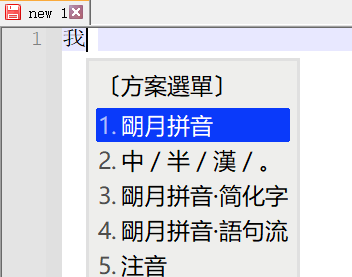
安装完成后，语言栏选择 小狼毫 输入法



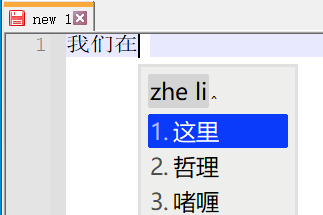
默认是拼音输入法：



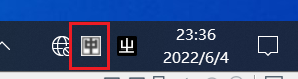
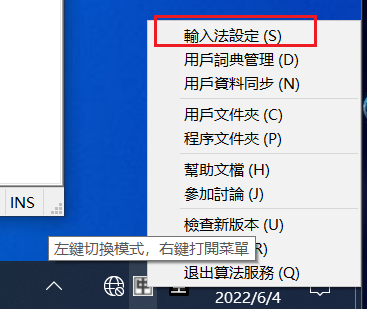
输入界面下 按下 F4 按键，可选择输入方案



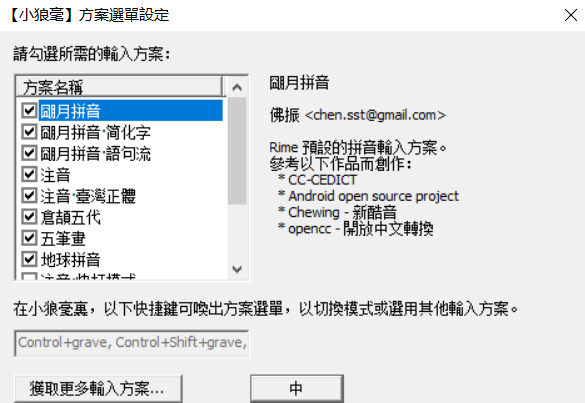
默认是 朙月拼音 方案，按下数字 3 ，可选择简体字版本的 月拼音 输入法方案，按下 ESC 按键可退出选择



右键单击任务栏右边 中字图标，选择 輪入法設定

可以进行一些简单的设置，如选择当前输入法方案以及显示框样式





**章一、基础配置信息**

windows 小狼毫weasel安装后，有2个主配置目录：

|  |  |
| --- | --- |
| 用户目录 | C:\Users\用户名\AppData\Roaming\Rime |
| 程序目录 | C:\Program Files (x86)\Rime\weasel-0.14.3 |

用户目录下的build子目录下存放最终生效的配置文件

C:\Users\cof\AppData\Roaming\Rime\build

用户目录下的**\*.userdb** 子目录为各输入法方案的用户数据存放目录

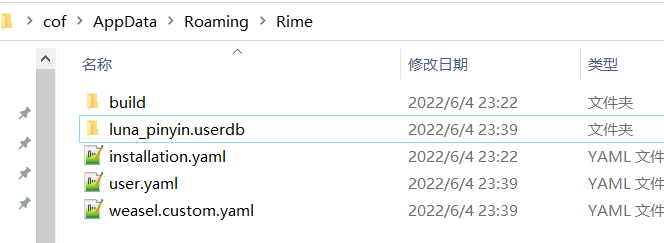
用户目录下的各.yaml文件为 各输入法方案配置文件，有：

xxx.schema.yaml #输入法方案定义文件，xxx一般同输入法方案名

xxx.dict.yaml #输入法方案词库文件，xxx一般同词库名

user.yaml #记录了当前用户的使用的输入法方案

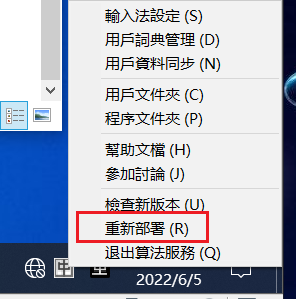
weasel.custom.yaml #记录了当前用户的一些个性配置，如输入框显示样式等



上图可见用户目录下默认没有\*.schema.yaml以及\*.dict.yaml等输入法配置文件，因为软件自带的输入法方案是内置的，不是用户自己创建的，所以在用户目录下没有，原始配置文件在安装目录下有（C:\Program Files (x86)\Rime\weasel-0.14.3\data）

**★安装其他输入法方案**

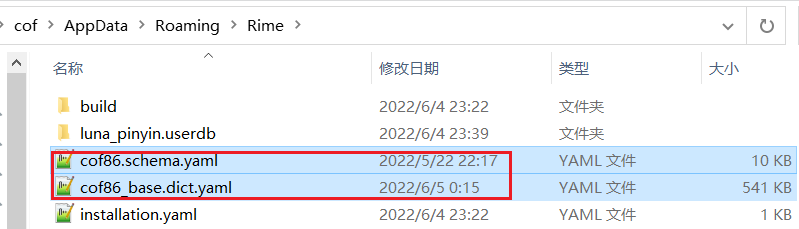
在用户目录下，把我们自已创建的输入法方案或者下载的输入法方案以及词库文件复制过来，再点击 重新部署 即安装完成，可再去选择使用即可



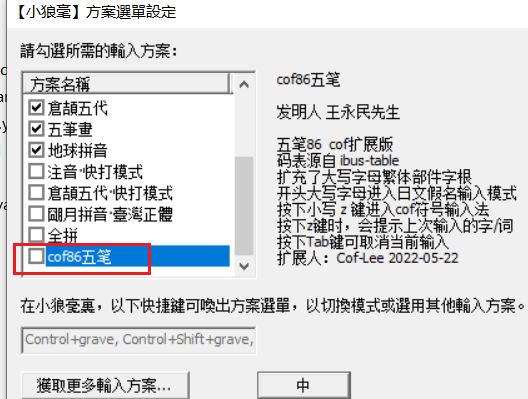
下载以下2个配置文件并复制到用户目录下 C:\Users\用户名\AppData\Roaming\Rime\

https://limaofu.github.io/scripts/cof86.schema.yaml

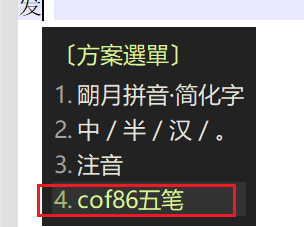
https://limaofu.github.io/scripts/cof86\_base.dict.yaml



重新部署 小狼毫，再打开 输入法设定，可见多了一个名为 cof86五笔 的方案



勾选它，点击“中”，选择一个显示样式，最后按F4，选择cof86五笔即可使用此输入法方案 进行打字了（可取消勾选不需要的输入方案）

★上面这个cof86五笔只是一个示例，词库只有基础单字，正式的可以下载github上的其他rime输入法方案，如：

https://github.com/limaofu/cof86 #作者自用的五笔86方案，源于rime-wubi词库

https://github.com/rime/rime-wubi #rime收集的五笔86方案

**章二、DIY自己的输入法方案**

在用户目录下，把我们自已创建的输入法方案及词库 文件复制过来，再点击 重新部署 即安装完成，再去选择使用即可。上一章已经做了个示例，现在从零开始diy自己的输入法方案

**①自定义schema方案**

首先，在用户目录下创建一个输入方案文件，如test.schema.yaml （文本文件，无txt后缀）

用文本编辑器打开，使用的文本编辑器最好是能显示出空格与Tab键的区别，内容如下：

# utf-8 #使用 # 井号作注释，可行内注释，rime的配置文件需要用utf8编码

# update: 2022-06-05

schema:

author:

- "发明人 cof先生" #作者

description: | #描述信息，以下可有多行字符串

test输入法

作者：xxx

创建日期：2022-06-05

name: "test输入法" #输入法显示名称

schema\_id: test #方案名称，全局唯一

version: 1.0 #版本号随便写

speller: #定义拼写规格的

delimiter: "/" #拼写分隔符为 /

alphabet: 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ' #拼写字母，可有数字与其他符号

max\_code\_length: 4 #最长编码字符数，五笔打一个字/词最多4个编码，拼音可以不限制

auto\_select: true #超出最长编码时，把前面的第一个候选项顶上屏

auto\_select\_unique\_candidate: true #在最大编码长度内，自动选择无重复候选的项 上屏

auto\_clear: max\_length #达到最大编码长度时，若无候选项则清空编码

switches:

- name: ascii\_mode #默认输入模式，0表示中文，1表示英文字母

reset: 1

states: ["中", "英"] #描述信息

- name: ascii\_punct #中文的 全角/半角 开关

reset: 0 # 默认0表示半角，1表示全角

states: ["。，", "．，"] #描述，0对应前面的字符串，1对应后面字符串

- name: full\_shape #英文符号的 全角/半角 开关，对中文符号无效

reset: 0 # 0表示半角，默认0

states: ["半", "全"] #描述信息

- name: extended\_charset #对Unicode字符过滤 开关

reset: 1 #1表示显示unicode扩展字符，扩A及以上

states: ["常utf", "扩utf"] # 0表示只显示Base0平面内的非扩展字符

- options: ["gbk", "utf8"] #显示的字符集 开关

reset: 1 #1表示显示unicode字符集，0表示只显示BGK字符集

states: ["Gbk", "Utf8"]

- name: zh\_trad #中文繁体字与简体字开关

reset: 0 #0表示关闭简体转繁体，即 显示简体字

states: ["关", "繁s2t"]

translator:

comment\_format: #候选框里候选字后面提示的编码字符 映射为 | 后面的字符

- "xlit|abcdefghijklmnopqrstuvwxyz|工了以在有地一上不是中国同民为这我的要和产发人经主z|"

- "xform/~/ -/" #把候选字后连接提示编码的 ~ 号转为 - 号

preedit\_format: #编码显示框里的编码字符映射

- "xform/^([a-z]\*)$/$1\\t[\\U$1\\E]/"

- "xlit|abcdefghijklmnopqrstuvwxyz|🐭🐮🐅🐰🐉🐘🐎🐏🐒🐓🐕🐷🐫🦌🐂👻👽💩❤💔🐥🐸🦖🦋🐝z|"

- "xlit|ABCDEFGHIJKLMNOPQRSTUVWXYZ|爿馬臣鹵𠀐車貝韋龍釒鳥門⺻飠糹訁☠|" #这里有些字符受操作系统字体的影响而没有显示出来

dictionary: cof86\_base #本输入法方案要用的 词库id

prism: cof86\_base #和词库id保持一致

disable\_user\_dict\_for\_patterns:

- "^z.\*$" #以 z 开头的编码不记入用户词库

enable\_charset\_filter: true #开启候选字字符集过滤

enable\_completion: true #开启编码提示，未输完编码的字/词 在后面提示剩余编码

enable\_encoder: false #关闭自动造词

enable\_sentence: true #允许打句子

enable\_user\_dict: false #是否开启用户词典，以记录动态字频和用户词词频（false关闭

encode\_commit\_history: false #不对已上屏的词自动整合成词

max\_phrase\_length: 12 #最大自动成词的词语长度

engine: #输入法加载的引擎

processors: #功能模块

- ascii\_composer

- recognizer

- chord\_composer

- key\_binder

- speller

- punctuator

- selector

- navigator

- express\_editor

segmentors:

- ascii\_segmentor

- matcher

- abc\_segmentor

- punct\_segmentor

- fallback\_segmentor

translators: #对候选字/词 的 增加/替换

- punct\_translator

- reverse\_lookup\_translator

- table\_translator

- history\_translator

filters: #对候选字/词的 过滤，filters 的功能是对 translators 翻译而来的候选项做修饰

- simplifier

- uniquifier

ascii\_composer:

good\_old\_caps\_lock: true #按下 Caps Lock键，则切换为大写字母

switch\_key:

Caps\_Lock: clear #按下Caps Lock键后，清空当前编码及候选字

Control\_L: noop #左Ctrl键，noop表示无反应

Control\_R: noop #右Ctrl键，noop表示无反应

Eisu\_toggle: clear #日文键盘的英数开关，clear表示清空当前编码及候选字

Shift\_L: commit\_code #左shift键，commit\_code表示把当前编码上屏（不是候选项）

Shift\_R: commit\_code #右shift键

key\_binder: #组合键设置某些快捷功能

bindings:

- {accept: semicolon, send: 2, when: has\_menu} #当有候选项时，按下;分号则发送字符2，即让第2个候选项上屏

- {accept: apostrophe, send: 3, when: has\_menu} #当有候选项时，按下'撇号则发送字符3，即让第3个候选项上屏

- {accept: space, send: space, when: has\_menu} #当有候选项时，按下空格键则发送空格字符，即第1候选项上屏

- {accept: space, send: Escape, when: composing} #当有空编码时，按下空格键则发送 Esc字符，即清空编码

- {accept: Tab, send: Escape, when: has\_menu} #当有候选项时，按下Tab键则发送Esc字符，即清空编码

- {accept: Tab, send: Escape, when: composing} #当有空编码时，按下Tab键则发送Esc字符，即清空编码

- {accept: "Shift+space", send: Down, when: has\_menu} #当有候选项时，按下Shift+空格，则发送down，选向下一个候选项

- {accept: "Control+f", toggle: zh\_trad, when: has\_menu} #对 zh\_trad 开关的切换

- {accept: minus, send: Page\_Up, when: has\_menu} #当有候选项时，按下-减号，则发送page\_up，候选项向上翻页

- {accept: equal, send: Page\_Down, when: has\_menu} #当有候选项时，按下=等号，则发送page\_down，候选项向下翻页

- {accept: Return, send: "Control+Shift+Return", when: has\_menu} #当有候选项时，按回车，则发送Ctrl+Shift+回车，候选项的编码提示 上屏

menu:

page\_size: 5 #候选框里的候选项数量，一页显示5行

alternative\_select\_labels: [ 🐷, 🐭, 🐰, 👽, 🐸, 🐷, 🐭, 🐰, 👽, 🐸] #候选框里候选项前的提示符默认是1，2，3，4，5 这些，改为指定的符号

#alternative\_select\_keys: "ABCDE" #选择候选项的字符，默认是1，2，3，4，5，如果1~5也做成了编码，则要改为其他字符

history:

input: 'z' #按下 z 键，显示历史上屏的候选项

size: 1 #历史上屏的候选项记录数

#initial\_quality: 1000

simplifier: #opencc滤镜设置

opencc\_config: s2t.json #滤镜配置文件，在程序目录的\data\opencc子目录下

option\_name: zh\_trad #滤镜开关名称，和上面switches下面的相对应

show\_in\_comment: true #true表示在原候选项后面作提示，不转换为候选项，false表示转为候选项

tips: all #all对所有字词都转换，char仅对单字进行转换

punctuator:

full\_shape: &punct #可让下面的half\_shape复用

" ": {commit: " "}

"!": {commit: "！"}

"\"": {pair: ["“", "”"]}

"#": {commit: "#"}

"$": ["$", "¥", "€", "£", "￥", "¢", "¤"]

"%": ["%", "°", "℃", "℉", "％"]

"&": {commit: "&"}

"'": {pair: ["‘", "’"]}

"(": "（"

")": "）"

"\*": ["\*", "＊", "★", "·", "・", "×", "※", "❂"]

"+": {commit: "+"}

",": {commit: "，"}

"-": {commit: "-"}

.: {commit: "。"}

"/": ["/"]

":": {commit: "："}

";": {commit: "；"}

"<": ["《", "『", "【", "〈", "«", "‹"]

"=": {commit: "="}

">": ["》", "』", "】", "〉", "»", "›"]

"?": {commit: "？"}

"@": ["＠", "@", "☯"]

"[": ["「", "【", "〔", "［"]

"\\": ["\\", "＼", "、"]

"]": ["」", "】", "〕", "］"]

"^": ["^", "……", "…"]

"\_": ["\_", "——"]

"`": ["`", "｀"]

"{": ["『", "〖", "｛"]

"|": ["·", "｜", "§", "¦"]

"}": ["』", "〗", "｝"]

"~": ["~", "～"]

half\_shape: \*punct #复用上面full\_shape定义的符号映射

symbols:

"/1": ["①", "⑴", "㈠"]

"/2": ["②", "⑵", "㈡"]

"/3": ["③", "⑶", "㈢"]

"/4": ["④", "⑷", "㈣"]

"/5": ["⑤", "⑸", "㈤"]

"/6": ["⑥", "⑹", "㈥"]

"/of": "℉"

"/oc": "℃"

recognizer: #对编码的识别，经过这一模块处理，才把编码去查询本输入方案的词库

patterns:

#email: "^[A-Za-z][-\_.0-9A-Za-z]\*@.\*$"

reverse\_lookup: "^z[a-zA-Z0-9\/]\*" #以z开头的编码作为反查编码，交给 reverse\_lookup 处理

punct: "^/([0-9]0?|[A-Za-z]+)$" #以 /开头及紧跟0~9的数字或字母，这样的编码识别为符号，交给 punctuator 处理

#uppercase: "[A-Z][-\_+.'0-9A-Za-z]\*$"

#url: "^(www[.]|https?:|ftp[.:]|mailto:|file:).\*$|^[a-z]+[.].+$"

reverse\_lookup:

dictionary: luna\_pinyin #反查的输入法方案

prefix: z #前缀为z，即第一个字符z不传给反查方案

suffix:

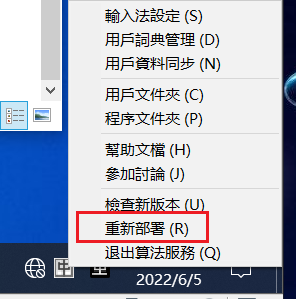
tips: "〔拼音反查〕" #反查时显示的 提示

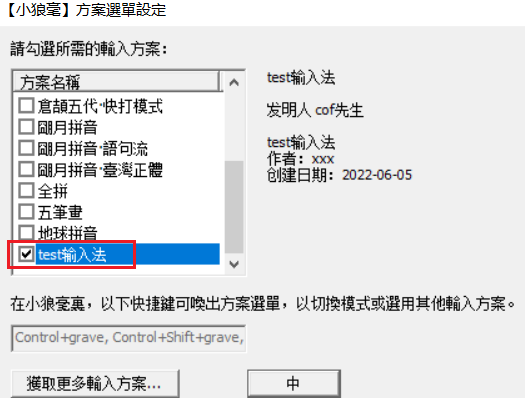
enable\_completion: true #反查时也提示未完成的编码

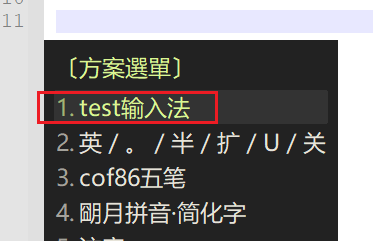
enable\_sentence: true #反查时也支持造句

#

#再点击 重新部署 即安装完成，可再去选择使用即可







dictionary: cof86\_base #schema配置文件指定了使用cof86\_base的词库，前面我们已经下载了 https://limaofu.github.io/scripts/cof86\_base.dict.yaml ，先用这个现成的词库试试效果，如下：

★正常打字（本例为五笔的词库）

，第一行为编码栏，字母已转为符号了

，第二行及以后为候选项，候选项后面的红色字符为编码提示符，也已转为其他符号了



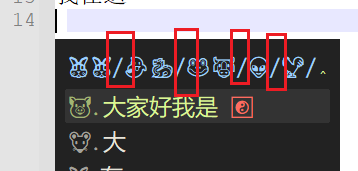
★反查效果（z开头的编码，后面的men为反查的编码，本例中反查去查了朙月拼音）



符号输入：



使用分隔符时（总编码）不受最长编码限制



**②自定义词库**

可以去开源世界找一个适合自己的词库，再拿来改造，增加或删除一些词条，以及修改词条的优先级，变成自己的词库

程序目录的/data子目录下有现成的一些方案的词库文件，可直接用来改造，如：

terra\_pinyin.dict.yaml #地球拼音

luna\_pinyin.dict.yaml #朙月拼音简体字词库

cangjie5.dict.yaml #仓颉5代词库

https://limaofu.github.io/scripts/wubi86\_jidian.dict.yaml #极点五笔词库

先自己创建一个dict词库文件如 test.dict.yaml ，熟悉一下原理再去改造，如果从0开始做自己的词库也是可以的，只是要添加常用的大概七，八万词条，是需要几年时间的。

词库内容如下：

# Rime dictionary: test

# encoding: utf-8

# author: Cof-Lee <cof8007@gmail.com>

---

name: test #词库名称，也是id，全局唯一

version: "1.0" #版本随便写

sort: by\_weight #候选项排序方式，以权重为优先级，权重越高越优先

# 即相同编码下，权重高的词条排在第一候选项，次高的排第二候选项）

columns: #词条（每一行）的各列含义，各列之间必须以Tab符分隔

- text #第一列为词条，候选项，可含空格字符（不可含Tab）

- code #第二列为该词条的编码（1到若干个编码字符）

- weight #权重，数字，4字节内的正整数，可省略，默认为0

- stem #造词码，可省略

encoder:

exclude\_patterns: #编码不含以z开头的字符

- '^z.\*$'

rules: #组词编码规则，要求方案允许自动造词，enable\_encoder为true

- length\_equal: 2 #2个字的组词编码规则

formula: "AaAbBaBb" #第一字的前2编码加第2字的前2编码

- length\_equal: 3 #3个字的组词编码规则

formula: "AaBaCaCb" #第一字的前1编码加 第2字的前1编码 加第3字前2编码

- length\_in\_range: [4, 10] #4到10个字的组词编码规则

formula: "AaBaCaZa" #前3字各取前1编码，最后一字取前1编码

...

工 a 7 aa

了 b 452 bn

子 b 451

以 c 33 ny

又 c 32

在 d 406 dh

大 d 405

有 e 64 de

月 e 63

地 f 59 fb

土 f 58

一 g 793 gg

王 g 792

上 h 83 hh

目 h 82

不 i 894 gi

水 i 893

是 j 407 jg

日 j 406

中 k 63 kh

口 k 62

国 l 59 lg

田 l 58

同 m 11 mg

山 m 10

民 n 53 na

已 n 52

为 o 71 yl

火 o 70

这 p 05 yp

之 p 804

我 q 1138 tr

金 q 1137

的 r 4000 rq

白 r 4999

要 s 494 sv

木 s 493

和 t 434 tk

禾 t 433

产 u 332 ut

立 u 331

发 v 307 nt

女 v 306

人 w 1710 ww

经 x 209 xc

主 y 558 yg

言 y 557

式 aa 73

节 ab 31

芭 ac 17

基 ad 520

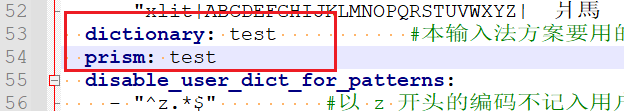
睯 nath 456

暋 natj 11300

测试 xxxx 3720

# 词条的各字段（列）分隔符必须为Tab制表符

然后将输入法schema文件里的dictionary以及prism的值改为自定义词库的名称 test



最后再重新部署，即可（输入xxxx可得如下候选项）



**③rime中的字符/按键名**

在key\_binder中可用

|  |  |  |  |
| --- | --- | --- | --- |
| BackSpace | 退格，键盘右上角的← | space | 空格 |
| Tab | 制表符，键盘左上角的→ | exclam | ! 感叹号 |
| Linefeed | 换行 | quote | ' 单引号 |
| Return | 回车 | quotedbl | " 双引号 |
| Pause | 暂停 | numbersign | # 序数番号，井号 |
| Break | 中断 | dollar | $ 美元符号 |
| Escape | ESC退出键 | percent | % 百分号 |
| Delete | delete键 | ampersand | & 和，与 |
| Home | home键 | apostrophe | ' 撇号，同单引号 |
| End | end键 | parenleft | ( 左括号 |
| Left | ← 左方向键 | parenright | ) 右括号 |
| Down | ↓ 下方向键 | asterisk | \* 星号 |
| Up | ↑ 上方向键 | plus | + 加号 |
| Right | → 右方向键 | comma | , 逗号 |
| Page\_Up | 上翻页键 | minus | - 减号 |
| Page\_Down | 下翻页键 | period | . 句号，小数点 |
| Shift | shift键，不分左右 | equal | = 等号 |
| Shift\_L | 左shift键 | greater | > 大于号 |
| Shift\_R | 右shift键 | less | < 小于号 |
| Control | Ctrl键，不分左右 | slash | / 斜杠 |
| Control\_L | 左Ctrl键 | colon | : 冒号 |
| Control\_R | 右Ctrl键 | semicolon | ; 分号 |
| Caps\_Lock | 大小写切换键 | question | ? 问号 |
| Menu | Menu键 | at | @ at号 |
| Alt | Alt键，不分左右 | asciicircum | ^ 插入号，数字6那里 |
| Alt\_L | 左Alt键 | bracketleft | [ 左方括号 |
| Alt\_R | 右Alt键 | bracketright | ] 右方括号 |
| underscore | \_ 下划线 | grave | ` 反撇号，重音符号，也叫backtick，键盘左上角与~波浪号同键 |
| braceleft | { 左花括号 | bar | | 管道符 |
| braceright | } 右花括号 | asciitidle | ~ 波浪号 |
| backslash | \ 反斜杠 |  |  |

**④自定义候选框样式**

首先在用户目录下的weasel.custom.yaml文件里添加自定义的候选框颜色style，内容如下：

customization: #这段就用默认的，不动它

distribution\_code\_name: Weasel

distribution\_version: 0.14.3

generator: "Weasel::UIStyleSettings"

modified\_time: "Mon Jun 22 18:02:55 2020"

rime\_version: 1.5.3

patch:

"preset\_color\_schemes/cof\_color":

author: cof

back\_color: 0x333333 #背景色，深灰，使用BGR颜色编码

border\_color: 0x009900 #边框颜色，黑色

candidate\_text\_color: 0xd8e3e6 #候选词条颜色

comment\_text\_color: 0xB8C419 #候选词后面的提示字符颜色

hilited\_back\_color: 0x003300 #?未生效

hilited\_candidate\_back\_color: 0xaaaaaa #选中的候选词那行的 背景色

hilited\_candidate\_text\_color: 0x3641FC #选中的候选词文字颜色

hilited\_comment\_text\_color: 0x88a409 #选中的候选词后面的提示字符颜色

hilited\_label\_color: 0x55dd00 #选中的候选词前面的label字符颜色

hilited\_text\_color: 0x03beec #选中的候选词颜色，未生效？

label\_color: 0x007700 #候选词前面的label字符颜色

name: cof\_color #样式名称，全局唯一

text\_color: 0x92f6da #文字颜色

"style/color\_scheme": cof\_color

#

再在test.schema.yaml输入法方案文件里，添加以下一段：

style:

color\_scheme: cof\_color #使用的颜色style

font\_face: "Serif\_Cof" #使用的字体，如果系统无此字体，则用系统默认字体

font\_point: 18 #字体大小，像素

label\_format: "%s " # %s表示menu里的labels字符，后面有个空格，表示候选项前的label字符与候选词条之间有个空格隔开

layout:

min\_width: 160 #候选框最小宽度，160像素

min\_height: 0 #候选框最小高度

border\_width: 3 #边框大小，3像素

margin\_x: 10

margin\_y: 10

spacing: 10 #编码行与下面候选词条之间的行间隙

candidate\_spacing: 5 #候选项之间的行空隙

hilite\_spacing: 5

hilite\_padding: 1

round\_corner: 15 #选中的候选词条4个边角圆角程度

#

最后，再重新部署，效果如下：

**⑤添加设置热键**

默认按下F4可打开Rime的输入法选择窗口，可设置其他的组合键

在程序目录/data子目录下的weasel.yaml文件里添加以下三行，可支持添加多个组合键

switcher:

hotkeys:

- "Control+grave"

重新部署即可

**章三、候选词条的过滤/转换**

**①多方案混输**

方案schema文件里：

engine: #输入法加载的引擎

translators: #对候选字/词 的 增加/替换

- table\_translator@其他方案名1

- table\_translator@其他方案名2

假如其他方案名为 luna\_pinyin ，再在schema配置下面添加一段：

luna\_pinyin:

dictionary: luna\_pinyin #前提是此方案及词库要存在

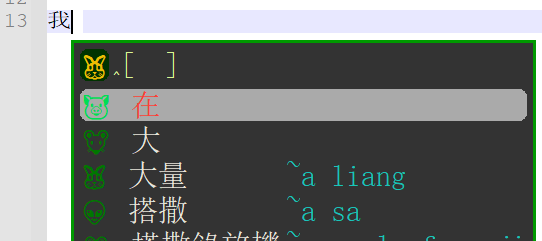
spelling\_hints: 9

enable\_completion: true

enable\_sentence: false

initial\_quality: -1

# 效果如下，本方案与luna\_pinyin方案可混输了



再添加个easy\_english的方案混输：

engine: #输入法加载的引擎

translators: #对候选字/词 的 增加/替换

- table\_translator@english

english:

dictionary: easy\_en #前提是此方案及词库要存在

spelling\_hints: 9

enable\_completion: true

enable\_sentence: false

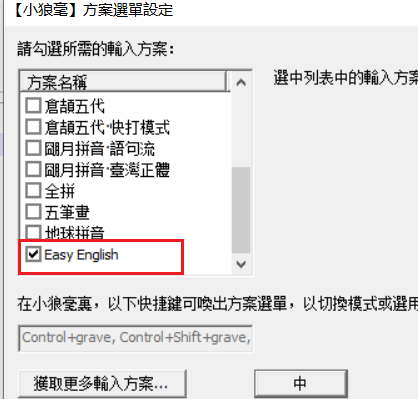
initial\_quality: -1

**# easy\_en方案及词库文件在：**

https://limaofu.github.io/scripts/easy\_en.dict.yaml

https://limaofu.github.io/scripts/easy\_en.schema.yaml

下载并复制到用户目录下，再重新部署，要勾选上Easy English方案



再使用test输入法方案，效果如下：



**②opencc滤镜对候选字转换**

rime软件自带有一些opencc的转换滤镜文件，就是简体字对应繁体字的映射文件

在程序目录 C:\Program Files (x86)\Rime\weasel-0.14.3\data\opencc 下面以.json结尾的文件为opencc滤镜文件，默认的有：

s2t.json #简体字转传统繁体字

s2tw.json #简体字转台湾风格的繁体字

s2hk.json #简体字转香港风格的繁体字

t2s.json #传统的繁体字转简体字

tw2s.json #台湾风格的繁体字转简体字

......

**简体字转繁体字示例：**

在test.schema.yaml里：

1.加开关：

switches:

- name: zh\_trad #开关名称

reset: 0 #设置为0表示关闭简体转繁体，即 显示简体字

states: ["关", "繁s2t"] #开关状态描述，前面对应0，后面对应1

2.加filters：

engine:

filters: #对候选字/词的 过滤

- simplifier

- uniquifier #此模块作用是如果候选项有多个相同的词条，则只显示其中第一个

3.指定使用的opencc滤镜文件

simplifier: #opencc滤镜设置

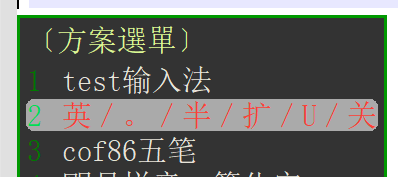
opencc\_config: s2t.json #滤镜配置文件，在程序目录的\data\opencc子目录下

option\_name: zh\_trad #滤镜开关名称，和上面switches下面的相对应

show\_in\_comment: true #true表示在原候选项后面作提示，不转换为候选项，false表示转为候选项

tips: all #all对所有字词都转换，char仅对单字进行转换

#最后重新部署，按F4进行开关的开启，



不过，开关太多了，描述字太少的话，根本不知道哪个开关对应zh\_trad

4.所以，最好是使用快捷键：

key\_binder: #组合键设置某些快捷功能

bindings: #增加下面这行

- {accept: "Control+f", toggle: zh\_trad, when: has\_menu} #对 zh\_trad 开关的切换

#最后重新部署，在使用test方案打字时，如果候选框有简体字，这时按下Ctrl加F键，则可见候选项后面有对应繁体字的提示，效果如下：



再按下Ctrl加F键，则提示关掉了

**举一反三：**

增加汉字对应的拼音滤镜，让候选项后面提示相应的拼音，效果如下：



首先下载拼音滤镜文件

https://limaofu.github.io/scripts/pinyin.json

https://limaofu.github.io/scripts/pinyin.txt

再复制到程序目录 C:\Program Files (x86)\Rime\weasel-0.14.3\data\opencc 下面

1.加开关：

switches:

- name: pinyin #开关名称

reset: 0 #设置为0表示关闭简体转繁体，即 显示简体字

states: ["关", "拼音"] #开关状态描述，前面对应0，后面对应1

2.加filters：

engine:

filters: #对候选字/词的 过滤

- simplifier

- "simplifier@pinyin"

- uniquifier

3.指定使用的opencc滤镜文件（增加下面一段）

pinyin: #opencc滤镜设置，全局第一级配置

opencc\_config: pinyin.json #滤镜配置文件，在程序目录的\data\opencc子目录下

option\_name: pinyin #滤镜开关名称，和上面switches下面的相对应

show\_in\_comment: true #true表示在原候选项后面作提示，不转换为候选项，false表示转为候选项

tips: all #all对所有字词都转换，char仅对单字进行转换

4.使用快捷键：

key\_binder: #组合键设置某些快捷功能

bindings: #增加下面这行

- {accept: "Control+p", toggle: pinyin, when: has\_menu} #对 pinyin 开关的切换

#最后重新部署，在使用test方案打字时，如果候选框有候选项，这时按下Ctrl加P键，则可见候选项后面有对应拼音的提示，效果如下：



**③候选项字符集过滤**

有时我们只想输出特定字符集里有的字符，可以使用charset\_filter模块

候选项只显示GBK字符集示例：

在test.schema.yaml里：

1.加开关：

switches:

- options: ["gbk", "utf8"] #这里是options，不是-name

reset: 0 #0表示只显示gbk字符集的候选项，按实际需要开启或关闭

states: ["Gbk", "Utf8"] #开关状态描述，前面对应0，后面对应1

2.加filters：

engine:

filters: #对候选字/词的 过滤

- simplifier

- charset\_filter@gbk

- uniquifier

#最后重新部署，在使用test方案打字时，只显示出在GBK字符集里的候选项

**④rime.lua脚本**

rime还支持lua脚本自定义translators和filters函数，在用户目录下创建一个名为rime.lua的文件，里面导出相应的函数，然后就可在engine/translators和engine/filters下面使用

**★增加实时日期及时间候选项**

rime.lua内容如下：

---------------

-- utf-8

-- 2022-06-05

-- 2个--表示注释

function date\_translator(input, seg)

-- 如果输入以下编码，则增加实时 日期 候选项

if (input == "date" or input == "rejj" or input == "jjad" or input == "now") then

-- 日期格式1，类似 2022年6月5日

num\_m=os.date("%m")+0

num\_m1=math.modf(num\_m)

num\_d=os.date("%d")+0

num\_d1=math.modf(num\_d)

date1=os.date("%Y年")..tostring(num\_m1).."月"..tostring(num\_d1).."日"

-- 候选项增加日期格式1

yield(Candidate("date", seg.start, seg.\_end, date1, " "))

-- 候选项增加此日期格式，如 2022-06-05

yield(Candidate("date", seg.start, seg.\_end, os.date("%Y-%m-%d"), " "))

end

-- 如果输入以下编码，则增加实时 时间 候选项

if (input == "time" or input == "jfuj") then

yield(Candidate("time", seg.start, seg.\_end, os.date("%H:%M"), " "))

yield(Candidate("time", seg.start, seg.\_end, os.date("%H:%M:%S"), " "))

yield(Candidate("time", seg.start, seg.\_end, os.date("%H点%M分"), " "))

yield(Candidate("time", seg.start, seg.\_end, os.date("%H点%M分%S秒"), " "))

end

end

# 再在test.schema.yaml里：

engine:

translators: #增加以下一行

- lua\_translator@date\_translator #@后面为rime.lua的函数名

再重新部署，输入 jjad或jfuj可得如下效果：



**★为候选项加上其所属字符集的注释**

在rime.lua文件中添加以下几个函数：

local charset = {

["cjk统一"] = { first = 0x4E00, last = 0x9FFF },

["拉丁补-1"] = { first = 0x0080, last = 0x00FF },

["拉丁扩a"] = { first = 0x0100, last = 0x017F },

["拉丁扩b"] = { first = 0x0180, last = 0x024F },

["私用区"] = { first = 0xE000, last = 0xF8FF },

["希腊"] = { first = 0x0370, last = 0x03FF },

["平仮名"] = { first = 0x3040, last = 0x309F },

["片仮名"] = { first = 0x30A0, last = 0x30FF },

["cjk部首补充"] = { first = 0x2E80, last = 0x2EFF },

["康熙字典部首"] = { first = 0x2F00, last = 0x2FDF },

["cjk标点符号"] = { first = 0x3000, last = 0x303F },

["cjk笔画"] = { first = 0x31C0, last = 0x31EF },

["注音"] = { first = 0x3100, last = 0x312F },

["朝鲜文兼容字母"] = { first = 0x3130, last = 0x318F },

["cjk兼容表意文字"] = { first = 0xF900, last = 0xFAFF },

["半角片仮名"] = { first = 0xFF61, last = 0xFF9F },

["扩A"] = { first = 0x3400, last = 0x4DBF },

["扩B"] = { first = 0x20000, last = 0x2A6DF },

["扩C"] = { first = 0x2A700, last = 0x2B73F },

["扩D"] = { first = 0x2B740, last = 0x2B81F },

["扩E"] = { first = 0x2B820, last = 0x2CEAF },

["扩F"] = { first = 0x2CEB0, last = 0x2EBEF },

["扩G"] = { first = 0x30000, last = 0x3134A },

["cjk兼容表意文字扩充"] = { first = 0x2F800, last = 0x2FA1F } }

--- -Compat Ext就是cjk兼容象形文字（表意文字）扩充

local function exists(single\_filter, text)

for i in utf8.codes(text) do

local c = utf8.codepoint(text, i)

if (not single\_filter(c)) then

return false

end

end

return true

end

local function is\_charset(s)

return function (c)

return c >= charset[s].first and c <= charset[s].last

end

end

function charset\_comment\_filter(input)

for cand in input:iter() do -- 使用 iter() 遍历所有输入候选项

for s, r in pairs(charset) do -- 判断当前候选内容 cand.text 中 文字属哪个字符集

if (exists(is\_charset(s), cand.text)) then

cand:get\_genuine().comment = cand.comment .. " " .. s .. " " .. string.format("%X",utf8.codepoint(cand.text))

break

end

end

yield(cand) -- 在结果中对应添加一个带注释的候选

end

end

# 再在test.schema.yaml里：

engine:

filters: #增加以下一行

- lua\_filter@charset\_comment\_filter #@后面为rime.lua的函数名

再重新部署，可见候选项后面有第一个字的所属unicdoe字符集分区提示：

