

CustomizationGuide

lotem edited this page on 9 Mar 2015 · 20 revisions


Rime 定製指南

必知必會

建議您在定製 Rime 輸入法之前瞭解 Rime 輸入方案的概念、Rime 中的數據文件分佈及作用等基礎知識。

必知必會

重新佈署的操作方法

- 【中州韻】點擊輸入法狀態欄上的  (Deploy) 按鈕 或：如果找不到狀態欄，在終端輸入以下命令，可觸發自動部署：

```
rm ~/.config/ibus/rime/default.yaml; ibus-daemon -drx
```

- 【小狼毫】開始菜單→小狼毫輸入法→重新佈署；當開啓托盤圖標時，右鍵點選「重新佈署」
- 【鼠鬚管】在系統語言文字選單中選擇「重新佈署」

對設置的修改於重新佈署後生效。編譯新的輸入方案需要一段時間，此間若無法輸出中文，請稍等片刻。

若部署完畢後，可以通過 `Ctrl+`` 喚出方案選單，輸入方案卻仍無法正常使用，可能是輸入方案未部署成功。請[查看日誌文件](#)定位錯誤。

查閱 DIY 處方集

已將一些定製 Rime 的常見問題、解法及定製檔鏈接俱收錄於下文的【DIY 處方集】

設定項速查手冊

[雪齋的文檔](#) 全面而詳細解釋了輸入方案及詞典中各設定項的含義及用法。

定製指南

Rime 輸入方案，將 Rime 輸入法的設定整理成完善的、可分發的形式。但並非一定要創作新的輸入方案，才可以改變 Rime 的行為。

當用戶需要對 Rime 中的各種設定做小幅的調節，最直接、但不完全正確的做法是：編輯用戶資料夾中那些 `.yaml` 文檔。

這一方法有弊端：

- 當 Rime 軟件升級時，也會升級各種設定檔、預設輸入方案。用戶編輯過的文檔會被覆寫為更高版本，所做調整也便丟失了。
- 即使在軟件升級後再手動恢復經過編輯的文件，也會因設定檔的其他部分未得到更新而失去本次升級新增和修復的功能。

因此，對於隨 Rime 發行的設定檔及預設輸入方案，推薦的定製方法是：

創建一個文件名的主體部份（「`.`」之前）與要定製的文件相同、次級擴展名（「`.yaml`」之前）為 `.custom` 的定製文檔：

```
patch:
  "一級設定項/二級設定項/三級設定項": 新的設定值
  "另一個設定項": 新的設定值
  "再一個設定項": 新的設定值
```

就是這樣：`patch` 定義了一組「補釘」，以源文件中的設定為基礎，寫入新的設定項、或以新的設定值取代現有設定項的值。

不懂？那看我來示範。

一例、定製每頁候選數

Rime 中，默認每頁至多顯示 5 個候選項，而允許的範圍是 1～9（個別 Rime 發行版可支持 10 個候選）。

設定每頁候選個數的默認值為 9，在用戶目錄建立文檔 `default.custom.yaml`：

```
patch:
  "menu/page_size": 9
```

重新佈署即可生效。

【注意】如果 `default.custom.yaml` 裏面已經有其他設定內容，只要以相同的縮進方式添加 `patch`：以下的部分，不可重複 `patch`：這一行。

若只需要將獨孤一個輸入方案的每頁候選數設為 9，以【明月拼音】為例，建立文檔 `luna_pinyin.custom.yaml` 寫入相同內容，重新佈署即可生效。

註：請參閱前文「重新佈署的操作方法」★

一例、定製標點符號

有的用家習慣以 `/` 鍵輸入標點「、」。

仍以【朙月拼音】爲例，輸入方案中有以下設定：

```
# luna_pinyin.schema.yaml
# ...

punctuator:
  import_preset: default
```

解釋：

`punctuator` 是 **Rime** 中負責轉換標點符號的組件。該組件會從設定中讀取符號映射表，而知道該做哪些轉換。

`punctuator/import_preset` 是說，本方案要繼承一組預設的符號映射表、要從另一個設定檔 `default.yaml` 導入。

查看 `default.yaml`，確有如下符號表：

```
punctuator:
  full_shape:
    # .....其他.....
    "/" : [ "/", " /", ÷ ]
    # .....其他.....
  half_shape:
    # .....其他.....
    "/" : [ " /", /, ÷ ]
    # .....其他.....
```

可見按鍵 `/` 是被指定到 `" /", /, ÷` 等一組符號了。並且全角和半角狀態下，符號有不同的定義。

欲令 `/` 鍵直接輸出「、」，可如此定製 `luna_pinyin.custom.yaml`：

```
patch:
  punctuator/full_shape:
    "/" : "、"
  punctuator/half_shape:
    "/" : "、"
```

以上在輸入方案設定中寫入兩組新值，合併後的輸入方案成爲：

```
# luna_pinyin.schema.yaml
# ...

punctuator:
  import_preset: default
  full_shape:
    "/" : "、"
```

```
half_shape:
  "/" : "、"
```

含義是、在由 `default` 導入的符號表之上，覆寫對按鍵 `/` 的定義。

通過這種方法，既直接繼承了大多數符號的默認定義，又做到了局部的個性化。

使用全套西文標點

有些用戶習慣在中文裏使用ASCII標點，那麼與其一個一個覆寫，不如 *整套都換掉*。

取得這份設定檔—— [Rime 別樣設定，使用西文標點](#) 在用戶資料夾保存為 `alternative.yaml` ；

再將輸入方案中的「導入 `default` 設定」通過打 `patch` 替換為「導入 `alternative` 設定」

```
# luna_pinyin.custom.yaml

patch:
  'punctuator/import_preset': alternative
```

就換上了自己習慣的一套標點！

一例、定製簡化字輸出

注意：

- 如果您只是需要 Rime 輸出簡化字，敲 `Ctrl+`` 組合鍵、從菜單中選擇「漢字→汉字」即可！
- 本例說明了其中原理，以及通過設定檔修改預設輸出字形的方法。

Rime 預設的詞彙表使用傳統漢字。這是因為傳統漢字較簡化字提供了更多信息，做「繁→簡」轉換能夠保證較高的精度。

Rime 中的過濾器組件 `simplifier`，完成對候選詞的繁簡轉換。

```
# luna_pinyin.schema.yaml
# ...

switches:
- name: ascii_mode
  reset: 0
  states: [ 中文, 西文 ]
- name: full_shape
  states: [ 半角, 全角 ]
- name: simplification # 轉換開關
  states: [ 漢字, 汉字 ]

engine:
  filters:
    - simplifier # 必要組件一
    - uniquifier # 必要組件二
```

以上是【朙月拼音】中有關繁簡轉換功能的設定。

在 `engine/filters` 中，除了 `simplifier`，還用了一件 `uniquifier`。這是因為有些時候，不同的候選會轉化為相同的簡化字，例如「鐘→钟」、「鍾→钟」。`uniquifier` 的作用是在 `simplifier` 執行轉換之後，將文字相同的候選項合併。

該輸入方案設有三個狀態開關：中 / 西文、全 / 半角、繁簡字。即 `switches` 之下三項。

每個開關可在兩種狀態（`states`）之間切換，`simplifier` 依據名為 `simplification` 的開關狀態來決定是否做簡化：

- 初始狀態下、輸出為傳統漢字、〔方案選單〕中的開關選項顯示為「漢字→汉字」。
- 選擇該項後、輸出為簡化漢字、〔方案選單〕中顯示「汉字→漢字」。
- `Rime` 會記憶您的選擇，下次打開輸入法時、直接切換到所選的字形。
- 亦可無視上次記住的選擇，在方案中重設初始值：`reset` 設為 0 或 1，分別選中 `states` 列表中的兩種狀態。

如果日常應用以簡化字為主：-（，則每每在〔方案選單〕中切換十分不便；於是佛振獻上默認輸出簡化字的設定檔：

```
# luna_pinyin.custom.yaml

patch:
  switches:                                # 注意縮進
    - name: ascii_mode
      reset: 0                             # reset 0 的作用是當從其他輸入方案切換到本方案時，
      states: [ 中文, 西文 ]              # 重設為指定的狀態，而不保留在前一個方案中設定的狀態。
    - name: full_shape                     # 選擇輸入方案後通常需要立即輸入中文，故重設 ascii_mode =
0;
      states: [ 半角, 全角 ]              # 而全 / 半角則可沿用之前方案中的用法。
    - name: simplification
      reset: 1                             # 增加這一行：默認啓用「繁→簡」轉換。
      states: [ 漢字, 汉字 ]
```

其實預設輸入方案中就提供了一套【朙月拼音】的簡化字版本，名為【簡化字】，以應大家“填表”之需。看他的代碼如何卻與上篇定製檔寫得不同：

```
# luna_pinyin.simp.schema.yaml
# ...

switches:
  - name: ascii_mode
    reset: 0
    states: [ 中文, 西文 ]
  - name: full_shape
    states: [ 半角, 全角 ]
  - name: zh_simp                          # 注意這裏（※1）
    reset: 1
    states: [ 漢字, 汉字 ]
```

```
simplifier:
  option_name: zh_simp      # 和這裏 (※2)
```

前文說，`simplifier` 這個組件會檢查名爲 `simplification` 的開關狀態；而這款【簡化字】方案卻用了一個不同名的開關 `zh_simp`，即※1處所示；並通過在※2行設定 `simplifier/option_name` 告知 `simplifier` 組件所需關注的開關名字。

何故？

還記得否，前文對「全 / 半角」這個開關的討論——當切換方案時，未明確使用 `reset` 重置的開關，會保持之前設定過的狀態。

【朙月拼音】等多數方案，並未重設 `simplification` 這個選項——因爲用戶換了一種輸入編碼的方式、並不意味着需要變更輸出的字形。

而【簡化字】這一方案不同，恰恰是表達變更輸出字形的需求；用戶再從【簡化字】切回【朙月拼音】時，一定是爲了「回到」繁體輸出模式。所以令【簡化字】使用獨立命名的開關、而非方案間共用的 `simplification` 開關，以避免影響其他輸入方案的繁簡轉換狀態。

一例、定製方案選單

在【小狼毫】方案選單設定介面上勾勾選選，就可以如此定製輸入方案列表：

```
# default.custom.yaml

patch:
  schema_list: # 對於列表類型，現在無有辦法指定如何添加、消除或單一修改某項，於是要在定製檔中
    將整個列表替換！
    - schema: luna-pinyin
    - schema: cangjie5
    - schema: luna-pinyin-fluency
    - schema: luna-pinyin-simp
    - schema: my-coolest-ever-schema # 這樣就啓用了未曾有過的高級輸入方案！其實這麼好的
      方案應該排在最前面哈。
```

無有設定介面時，又想啓用、禁用某個輸入方案，手寫這樣一份定製檔、重新佈署就好啦。

一例、定製喚出方案選單的快捷鍵

喚出方案選單，當然要用鍵盤。默認的快捷鍵爲 `Ctrl+`` 或 `F4`。

不過，有些同學電腦上 `Ctrl+`` 與其他軟件衝突，`F4` 甚至本文寫作時在【鼠鬚管】中還不可用。又或者有的玩家切換頻繁，想定義到更好的鍵位。

那麼……

```
# default.custom.yaml
```

```
patch:
  "switcher/hotkeys": # 這個列表裏每項定義一個快捷鍵，使哪個都中
    - "Control+s" # 添加 Ctrl+s
    - "Control+grave" # 你看寫法並不是 Ctrl+` 而是與 IBus 一致的表示法
    - F4
```

按鍵定義的格式為「修飾符甲+修飾符乙+...+按鍵名稱」，加號為分隔符，要寫出。

所謂修飾符，就是以下組合鍵的狀態標誌或是按鍵彈起的標誌：

- Release——按鍵被放開，而不是按下
- Shift
- Control
- Alt——Windows上 Alt+字母 會被系統優先識別為程序菜單項的快捷鍵，當然 Alt+Tab 也不可
- 嗯，Linux 發行版還支持 Super, Meta 等組合鍵，不過最好選每個平臺都能用的啦

按鍵的名稱，大小寫字母和數字都用他們自己表示，其他的按鍵名稱參考這裏 X11/keysymdef.h 的定義，去除代碼前綴 XK_ 即是。

一例、定製【小狼毫】字體字號

雖與輸入方案無關，也在此列出以作參考。

```
# weasel.custom.yaml

patch:
  "style/font_face": "明兰" # 字體名稱，從記事本等處的系統字體對話框裏能看到
  "style/font_point": 14 # 字號，只認數字的，不認「五號」、「小五」這樣的
```

一例、定製【小狼毫】配色方案

註：這款配色已經在新版本的小狼毫裏預設了，做練習時，你可以將文中 `starcraft` 換成自己命名的標識。

```
# weasel.custom.yaml

patch:
  "style/color_scheme": starcraft # 這項用於選中下面定義的新方案
  "preset_color_schemes/starcraft": # 在配色方案列表裏加入標識為 starcraft 的新方案
    name: 星際我爭霸 / StarCraft
    author: Contralisk <contralisk@gmail.com>, original artwork by Blizzard Entertainment
    text_color: 0xccaa88 # 編碼行文字顏色，24位色值，用十六進制書寫方便些，順序是藍綠紅0xBBGGRR
    candidate_text_color: 0x30bb55 # 候選項文字顏色，當與文字顏色不同時指定
    back_color: 0x000000 # 底色
    border_color: 0x1010a0 # 邊框顏色，與底色相同則為無邊框的效果
    hilited_text_color: 0xfecb96 # 高亮文字，即與當前高亮候選對應的那部份輸入碼
    hilited_back_color: 0x000000 # 設定高亮文字的底色，可起到凸顯高亮部份的作用
```

```
hilitied_candidate_text_color: 0x60ffa8 # 高亮候選項的文字顏色，要醒目！
hilitied_candidate_back_color: 0x000000 # 高亮候選項的底色，若與背景色不同就會顯出光
```

棒

效果自己看！

DIY 處方集

已將一些定製 Rime 的常見問題、解法及定製檔鏈接收錄於此。

建議您首先讀完《定製指南》、通曉相關原理，以正確運用這些處方。

初始設定

在方案選單中添加五筆、雙拼

<https://gist.github.com/2309739>

做此例，可啓用任一預設或自訂輸入方案，如【粵拼】、【注音】等。（詳解：參見前文「定製方案選單」一節）

如果下載、自己製作了非預設的輸入方案，將源文件複製到「用戶資料夾」後，也用上面的方法將方案標識加入選單！

修改於重新佈署後生效。

【小狼毫】外觀設定

上文已介紹設定字體字號、製作配色方案的方法。

使用橫向候選欄、嵌入式編碼行：

```
# weasel.custom.yaml
patch:
  style/horizontal: true      # 候選橫排
  style/inline_predit: true  # 內嵌編碼（僅支持TSF）
  style/display_tray_icon: true # 顯示托盤圖標
```

【鼠鬚管】外觀與鍵盤設定

鼠鬚管從 0.9.6 版本開始支持選擇配色方案，用 `squirrel.custom.yaml` 保存用戶的設定。

<https://gist.github.com/2290714>

在特定程序裏關閉中文輸入

【鼠鬚管】0.9.9 開始支持這項設定：

在指定的應用程序中，改變輸入法的初始轉換狀態。如在

- 終端 Terminal / iTerm
- 代碼編輯器 MacVim
- 快速啓動工具 QuickSilver / Alfred 等程序裏很少需要輸入中文，於是鼠鬚管在這些程序裏默認不開啓中文輸入。

自定義 Mac 應用程序的初始轉換狀態，首先查看應用的 `Info.plist` 文件得到 該應用的 `Bundle Identifier`，通常是形如 `com.apple.Xcode` 的字符串。

例如，要在 Xcode 裏面默認關閉中文輸入，又要在 Alfred 裏面恢復開啓中文輸入，可如此設定：

```
# example squirrel.custom.yaml
patch:
  app-options/com.apple.Xcode:
    ascii_mode: true
  app-options/com.alfredapp.Alfred: {}
```

註：一些版本的 Xcode 標識爲 `com.apple.dt.Xcode`，請注意查看 `Info.plist`。

【小狼毫】0.9.16 亦開始支持這項設定。

例如，要在 gVim 裏面默認關閉中文輸入，可如此設定：

```
# example weasel.custom.yaml
patch:
  app-options/gvim.exe: # 程序名字全用小寫字母
    ascii_mode: true
```

輸入習慣

使用**Control**鍵切換中西文

<https://gist.github.com/2981316>

以及修改Caps Lock、左右Shift、左右Control鍵的行爲，提供三種切換方式。詳見 Gist 代碼註釋。

方便地輸入含數字的西文用戶名

通常，輸入以小寫拉丁字母組成的編碼後，數字鍵的作用是選擇相應序號的候選字。

假設我的郵箱地址是 `rime123@company.com`，則需要在輸入rime之後上屏或做臨時中西文切換，方可輸入數字部分。

爲了更方便輸入我的用戶名 `rime123`，設置一組特例，將 `rime` 與其後的數字優先識別西文：

<https://gist.github.com/3076166>

以方括號鍵換頁

<https://gist.github.com/2316704>

添加 Mac 風格的翻頁鍵 []。這是比較直接的設定方式。下一則示例給出了一種更系統、可重用的設定方式。

使用西文標點兼以方括號鍵換頁

<https://gist.github.com/2334409>

詳見上文「使用全套西文標點」一節。

以回車鍵清除編碼兼以分號、單引號選字

<https://gist.github.com/2390510>

適合一些形碼輸入法（如五筆、鄭碼）的快手。

關閉逐鍵提示

`table_translator` 默認開啓逐鍵提示。若要只出精確匹配輸入碼的候選字，可關閉這一選項。

以【倉頡五代】爲例：

```
# cangjie5.custom.yaml
patch:
  translator/enable_completion: false
```

關閉用戶詞典和字頻調整

以【五筆86】爲例：

```
# wubi86.custom.yaml
patch:
  translator/enable_user_dict: false
```

關閉碼表輸入法連打

註：這個選項僅針對 `table_translator`，用於屏蔽倉頡、五筆中帶有太極圖章「☯」的連打詞句選項，不可作用於拼音、注音、速成等輸入方案。

以【倉頡】爲例：

```
# cangjie5.custom.yaml
patch:
  translator/enable_sentence: false
```

關閉倉頡與拼音混打

默認，給出倉頡與拼音候選的混合列表。

如此設定，直接敲字母只認作倉頡碼，但仍可在敲`之後輸入拼音：

```
# cangjie5.custom.yaml
patch:
  abc_segmentor/extra_tags: {}
```

空碼時按空格鍵清空輸入碼

首先需要關閉碼表輸入法連打（參見上文），這樣才可以在打空時不出候選詞。

然後設定（以五筆86為例）：

```
# wubi86.custom.yaml
patch:
  translator/enable_sentence: false
  key_binder/bindings:
    - {when: has_menu, accept: space, send: space}
    - {when: composing, accept: space, send: Escape}
```

模糊音

【朗月拼音】模糊音定製模板

<https://gist.github.com/2320943>

【明月拼音・簡化字 / 臺灣正體 / 語句流】也適用，只須將模板保存到

`luna_pinyin_simp.custom.yaml`、`luna_pinyin_tw.custom.yaml` 或
`luna_pinyin_fluency.custom.yaml`。

對比模糊音定製模板與【朗月拼音】[方案原件](#)，可見模板的做法是，在 `speller/algebra` 原有的規則中插入了一些定義模糊音的代碼行。

類似方案如雙拼、粵拼等可參考模板演示的方法改寫 `speller/algebra`。

【吳語】模糊音定製模板

<https://gist.github.com/2015335>

編碼反查

設定【速成】的反查碼為粵拼

<https://gist.github.com/2944320>

設定【倉頡】的反查碼為雙拼

<https://gist.github.com/2944319>

在Mac系統上輸入emoji表情

參考 <https://gist.github.com/2309739> 把 emoji 加入輸入方案選單；

切換到 emoji 輸入方案，即可通過拼音代碼輸入表情符號。[查看符號表](#)

輸入 all 可以列出全部符號，符號後面的括弧裏標記其拼音代碼。

若要直接在【朙月拼音】裏輸入表情符號，請按此文設定：

<http://gist.github.com/3705586>

五筆簡入繁出

【小狼毫】用家請到[下載頁](#)取得「擴展方案集」。

安裝完成後，執行輸入法設定，添加【五筆·簡入繁出】輸入方案。

其他版本請參考這篇說明：

<https://gist.github.com/3467172>

修正不對稱繁簡字

繁→簡即時轉換比簡體轉繁體要輕鬆許多，卻也免不了個別的錯誤。

比如這一例，「乾」字是一繁對多簡的典型。由它組成的常用詞組，openccc 都做了仔細分辨。但是遇到較生僻的詞組、專名，就比較頭疼：

<http://tieba.baidu.com/p/1909252328>

活用標點創建自定義詞組

在【朙月拼音】裏添加一些自定義文字、符號。可以按照上文設定「emoji表情」的方式為自定義詞組創建一個專門的詞典。

可是建立詞典稍顯繁瑣，而活用自定義標點，不失為一個便捷的方法：

```
# luna_pinyin.custom.yaml
# 如果不需要 ` 鍵的倉頡反查拼音功能，則可利用 ` 鍵輸入自定義詞組
patch:
  recognizer/patterns/reverse_lookup:
    'punctuator/half_shape/':
      - '佛振 <chen.sst@gmail.com>'
      - 'http://code.google.com/p/rimeime/'
      - 上天賦予你高的智商，教你用到有用的地方。
```

上例 recognizer/patterns/reverse_lookup: 作用是關閉`鍵的反查功能。若選用其他符號則不需要這行。又一例：

```
patch:
  'punctuator/half_shape/+' : '+_+'

```

'punctuator/half_shape/+' 因為字符串包含符號，最好用 單引號 括起來，避免符號的轉義問題。

重定義「/」這個符號，無法用上面演示的路徑連寫方式，那就分開寫：

```
patch:
  punctuator/half_shape:
    '/': [ '/', '/hello', '/bye', '/* TODO */' ]
    '+': '+_+'

```

+ Add a custom footer