LinuxDoc+Emacs+Ispell-HOWTO 中文版

作者： Philippe MARTIN (*feloy@wanadoo.fr*)
英譯者： Sébastien Blondeel (*Sebastien.Blondeel@lifl.fr*)
中譯者：李安珊 An-Shan Lee (*anshan@hello.com.tw*)
v0.4, 27 February 1998

這份文件主要是提供給 Linux HOWTOs 的作者與譯者及其他參與 Linux 文件計劃 (Linux Documentation Project) 者參考使用。從本文大家可以找到一些有關使用Emacs 與 Ispell 工具的提示。

目錄

1 前文

1.1 版權

1.2 銘謝

特別感謝 Sébastien Blondeel。 Sébastien 是個非常囉哩吧嗦的人，不停的問我一大堆有關 Emacs 安裝的問題。:-) 但也由於他所問的問題都很有意義，才使我能更深切的了解 Emacs，並將我所學到的知識發表出來，提供給大家參考。

1.3 評論

如果您有任何建議可以改善本文，請不要猶豫盡快的告訴我，我會仔細的參考您的意見。
對於此文件所探討的主題，如果您有任何問題的話，也請盡快的告訴我，我會非常樂意的回答您，因爲您的發問或許也能幫助我改善此文件。[1]

1.4 版本

這份文件與下列版本相關:

- Sgml-tools 版本 0.99，
- Emacs 版本 19.34，

[1]英譯者註：假如大家認爲這文件的英文很難懂的話，那就怪我翻譯的不好。
中譯者註：那如果大家對這中文文件有疑問的話，那還是怪罪英譯者囉，因爲我是照他的英文翻譯過來的。:-P
開玩笑的，如果大家認爲這份文件中文翻的不好，請多多包涵，並提供意見給我，以便我改善它，謝謝。

- Ispell 版本 3.1，
- 本文舉例內相關的 Emacs 程式庫都與上列的 Emacs 版本一起發行， `iso-sgml` 則與 XEmacs 一起發行，而 `psgml` 則是獨自發行的程式庫。

2 簡介

2.1 SGML

Standard Generalised Mark-up Language，或稱爲 **SGML**，是一種可以定義文件形態的語言。

譬如，您可以定義食譜的文件型態，第一部先呈現其主成分，第二部介紹佐料，第三部則一步一步的指示如何烘烤蛋糕，之後一幅好看的圖畫就可表現其可口美味了。

這就是所謂的文件形態定義 *(Document Type Definition)*。它並沒有定義最後的產品應是什麼樣子，它只說明文件的內容大約包括了什麼。

以此類推，您在讀了我這食譜的型態定義的舉例之後，您的腦子裡就會有您自己類似的食譜，或者您想的正是您最喜歡的廚子所呈現給您的，對吧？然而，他們其實看起來都是不同的：在我的想像裡，我的成果照片是放在浴室櫃子的左上角，而主成分的表單是在後花園裡，介於游泳池與烤肉區之間。您的呢？

多謝 SGML 的標準定義，我們可以按照其所提供的格式輕鬆的寫一份文件，而不用擔心最後讀者會看不懂我們在寫些什麼。

2.2 `LinuxDoc` 型態定義

正如您可能猜到的，這型態適用於寫作關於 Linux 的文件。

此種文件通常建構如下：文件的開始是標題，接著是作者的名稱，文件的版本號碼與日期。再來是摘要（這樣讀者就不用在瀏覽過整篇文件之後才知道這並不是他們想看的文件），之後是目錄，顯示文件結構，這樣一來那些急於想知道某一特定內容的人就可直接跳到那部分去讀。

然後就是一系列的章節，段落等。在這部分，您可以插入部分程式，改變字體來強調某個字眼或句子，或插入列表 (lists)，以導引讀者至文件的另一部份讀取相關資料，等等。

要寫這種文件，您只需要正確的詳細說明文件的標題，作者，日期，版本，章節部分，及大約說明何時將插入某一列表 (list)，其包含的元件是什麼等等即可。

2.3 `SGML-Tools`

SGML-Tools可以將一文件的特殊規格轉換成您所想要的格式。假如您要將此文件存在您個人的程式庫，您可以選擇轉成 *PostScript* 格式。假如您要將它透過網路與世界分享，可將它轉成 *HTML* 格式。假如您必須在 Windows 下讀取此份文件，您可以把它轉成 *RTF* 格式，並使用文書軟體讀取。或者您也可以將之轉成上述所有格式，以便您任何時候使用。

您可從這 FTP 站 *ftp://ftp.lip6.fr/pub/sgml-tools/* 取得 SGML-Tools。

3 您的第一份文件

3.1 從一份純文字檔文件開始

假如您要將一純文字檔文件轉成 SGML 以便再將之轉成其他的格式，做法如下：

1. 再最開頭時加上下列字句與標籤：

```
<!doctype linuxdoc system>
 <article>
  <title>標題 (Title) 放此</title>
  <author>
   作者，作者的電子郵件帳號，等等。
  </author>
  <date>
   版本及日期
  </date>
```

2. 如果您要在開頭時摘要此文件的內容，請將下列標籤用於此段落的頭尾： `<abstract>` 與 `</abstract>`。
3. 之後插入這 `<toc>` 標籤， SGML-Tools 自會顯示文件的目錄 *(Table Of Contents)*。
4. 再來，在每一章 (chapter) 的開始，以下列字句與標籤取代每一章的號碼及標題：

```
<sect>這一章的標題 (The Title Of The Chapter)
```

   再於這一章的結尾加上此 `</sect>` 標籤。
   註： 您不需放上這一章的相對號碼， SGML-Tools 自會幫您加上。
5. 以相同的方法用於接下來的分節 (section)。您必須刪除每一節的號碼，並以此 `<sect1>` 標籤標於每一節開頭的標題，及在每一節的結尾標上 `</sect1>`。
6. 在每一節中，您也可以使用 `<sectn>` 與 `</sectn>` 標籤，將之分層多達 4 個層次，每一層使用同樣的標籤，其中 n=2,3, 或 4，依此類推。
7. 之後，在每一段落 (paragraph) 的開始，插入 `<p>` 標籤。
8. 在段落中，如果您需要強調某些部分，就使用 `<it>` 與 `</it>` (斜體)， `<bf>` 與 `</bf>` (粗體)，或 `<tt>` 與 `</tt>` (打字體) 標籤。
9. 若要插入列表 (list)，請依下列方式：

```
例子：這是一個四行的列表：

 - 第一行放在這裡
 - 跟著第二行
 - 再來第三行
 - 第四行。
```

請以下列方式來標明上列列表：

```
 這是一個四行的列表：
<itemize>
<item>第一行放在這裡
<item>跟著第二行
<item>再來第三行
<item>t第四行。
</itemize>
```

10. 當您要在段落中放進某程式的一部分，或者需要突顯某些地方，請依下列方式：

```
<verb>
10 REM 喔，老天！這是什麼？
20 REM 我以爲這東西早就不見了！
30 PRINT "我回到...";
40 PRINT "拯救世界。"
50 INPUT "你從哪得知的？ ",M$
60 IF M$="阿呆" THEN PRINT "你很聰明。":GOTO PARADISE
70 ELSE PRINT "雖然你仍是傻呼呼的。":GOTO RICHMOND
</verb>
```

11. 到目前爲止，您所學到的 SGML 格式技術已經相當夠用。假如您想再加強您的文件，您可以參考 **SGML-Tools** 的使用指引，裡面詳盡的介紹了有關 **LinuxDoc** 文件型態的應用。

4 設定 Emacs

4.1 重音文字 (Accented Characters)

假如您需要以法文或其他西歐語言來寫文件，您必須使用 8-bit 文字 (8-bit characters)，才可顯示其有重音的文字。這樣的設置才可以令 Emacs 接受這些文字。

4.1.1 顯示 8-bit 文字

要令 Emacs 顯示 8-bit 文字，您必須將下列指令加入您的 `.emacs` 檔案：

```
(standard-display-european 1)
(load-library "iso-syntax")
```

假如您在一個不支援 8-bit 的終端機上使用 Emacs，您可以使用 `iso-ascii` 程式庫 (`(load-library "iso-ascii")`)，它會指令 Emacs 以最相似的形式來顯示這些文字。

4.1.2 8-bit 文字的鍵入

假如您的鍵盤允許您鍵入有重音的文字，那就沒有問題，要不然也可照下列方式來補救：

`Iso-acc` 程式庫　Emacs `iso-acc` 程式庫可以讓您使用 7-bit 的鍵盤鍵入 8-bit 的文字。
要將下列指令加入您的 `.emacs` 檔案：

```
(load-library "iso-acc")
```

然後在使用 Emacs 時，開啓您所需要編輯的檔案，鍵入 `Meta-x iso-accents-mode`。
這樣您若要鍵入法文 *café* 的 **é** 的話，您就可以先鍵入，然後再鍵入 `e`。總之，您需先鍵入那個文字的重音，之後再鍵入相對的字母（不論大寫或小寫）。以下是一些您可以使用的重音：

- `'` : Acute
- `` ` `` : Grave
- `^` : Circumflex
- `"` : Dieresis
- `~` : Tilde, cedilla, 與其他特別的例子 (cf iso-acc.el).
- `/` : 在一字母上加線，等等。

如果您只需要這些重音符號之一，但不是相對的重音字母，那就在符號之後鍵入一空白鍵。例如，要輸出 *l'éléphant*，就鍵入 `l`，⟨空白⟩，`e l`，`e ...` 即可。
您可以在 `iso-acc.el` 檔案內找到所有您需要的重音組合。

⟨*Meta*⟩ 鍵　可以在某些終端機上使用 ⟨*Meta*⟩ (或 ⟨*Alt*⟩) 鍵來鍵入 8-bit 文字。例如，您可以鍵入 ⟨*Meta*⟩**-i** 來輸出 **é** 這個字。
但是就我所知， Emacs 將 ⟨*Meta*⟩ 鍵用於其他場合，其他的程式庫似乎也不提供這種重音輸入方式。
不過我們還是可以以下列方式來補救：

```
(global-set-key "\ei" '(lambda () (interactive) (insert ?\351)))
                   -                                  ---
```

只要將上列指令加入您的 `.emacs` 檔案內，您就可以用 ⟨*Meta*⟩**-i** 來輸出 **é**。您也可以重新定義您要的組合，只要您用您要的字鍵 (key) 取代 **i**，用正確的編碼 (code) 取代 **351**，再配對起來，就可以得到您要的重音字母。您可參考 ISO-8859-1 character set 取得您要的編碼。
注意！ 某些 local modes 或許會重新定義您的字鍵組合。

4.1.3 8-bit SGML 文字的顯示

使用 SGML 時，您可以利用巨集 (macros) 來輸入重音文字。例如，要輸入 é 就可用 **é**。但通常，需要讀取 SGML 的應用軟體都可以讀取 8-bit 文字，所以一般都用不到這種巨集。或許有些軟體仍然無法讀取 8-bit 文字，那麼這時您就可以利用上述巨集了。

般時候，在使用 Emacs 時，您可以利用 `iso-sgml` 程式庫來輸入重音文字。當您要將文件檔案存到磁碟時，它也會幫您把 8-bit 文字轉成 SGML 的格式，以方便您讀取。

所以有了這個程式庫，您就可以在使用 Emacs 時，輕鬆的輸入或讀取 8-bit 文字，甚至一個不接受 8-bit 文字的軟體，也可以讀取您編寫的 8-bit 文件。

您只要將下列文字加入 `.emacs` 檔案內，您就可以使用上述程式庫：

```
(setq sgml-mode-hook
'(lambda () "Defaults for SGML mode."
 (load-library "iso-sgml")))
```

4.2 SGML 模式 (SGML mode)

當載入 **.sgml** 相關文件時， Emacs 通常會自動進入 **sgml** 模式。 假如沒有的話，您可以鍵入 `Meta-x sgml-mode` 來驅動它進入 **sgml** 模式。或者將下列指令加入 `.emacs` 檔案內來設定它自動進入：

```
(setq auto-mode-alist
(append '(("\.sgml$"  . sgml-mode))
         auto-mode-alist))
```

在 sgml 模式下，您可以選擇要如何輸出 8-bit 文字。比如，使用 `Meta-x sgml-name-8bit-mode`（或是選單項目上的 *SGML/Toggle 8-bit insertion*），您可以選擇直接輸出 8-bit 文字的格式，或是輸出 SGML 的格式（即 **&...;** 格式）。

在此模式下，您也可以使用 `Meta-x sgml-tags-invisible`（或是選單項目上的 *SGML/Toggle Tag Visibility*） 來選擇是否要隱藏或顯示 SGML 標籤。

4.3 PSGML 模式 (PSGML mode)

PSGML 模式對於在使用 Emacs 下編輯 SGML 文件很有助益。

這份文件 *psgml-linuxdoc* 解釋了如何安裝這個模式，及如何將之應用在 *LinuxDoc*。

4.4 其他相關模式

4.4.1 Auto-fill mode

在一般模式下，當您打完一個段落，您通常會鍵入 ⟨*Return*⟩ 以便換到下一行，否則您文章的段落就會相連，無法分清楚。但當您使用 ⟨*Return*⟩ 以便換到下一行時，您的段落的右邊通常是參差不齊的。

但假如您不換到下一行而繼續鍵入文字，在某些編輯器下，您就無法看到右邊邊界外的文字是什麼。

在這情形下，**auto-fill** 模式可以幫您改正這情況。當您鍵入超過預設值的第70列時，在這模式下，您就會自動換到下一行。

列是執行 auto-fill 模式的指令，並可將一行的寬度調整爲80列：

```
(setq sgml-mode-hook
      '(lambda () "Defaults for SGML mode."
          (auto-fill-mode)
          (setq fill-column 80)))
```

5 Ispell

如果您要在使用 Emacs 時作拼字檢查，您可以使用 **Ispell** 套件及其 Emacs 模式。

5.1 選擇您的預設辭典 (default dictionaries)

您可以設定 Emacs，使之在開始載入某一文件時，便自動選擇所需的辭典（您可以選擇好幾個）。預設辭典 (main dictionary) 即主要辭典，是與 Ispell 一起發行的辭典。您可以選擇您所需要的語言。第二個即是您的個人辭典 (personal dictionary)，您可以設定要 Ispell 將您在主要辭典裡所找不到的字放到這裡。

假如您要將 Ispell 裡的法文辭典設爲預設辭典，並想用您主目錄 (home directory) 下的 `.ispell-dico-perso` 檔案作爲個人辭典，可以在 `.emacs` 檔案裡加入下列指令：

```
(setq sgml-mode-hook
'(lambda () "Defauts for SGML mode."
(setq ispell-personal-dictionary "~/.ispell-dico-perso")
(ispell-change-dictionary "francais")
))
```

5.2 爲特定文件設定個別的辭典

如果您沒有用同一語言來檢查整份文件的拼字，那麼您可能會遇到一些困難。但當您在翻譯文章時，您很有可能經常交換使用不同的語言與辭典。

不管是自動設定的或是手動的，我並不清楚 Lisp 是如何選擇所需的主辭典和個人辭典。（如果您知道的話，請來信告訴我！）

不過呢，在每一個檔案的結尾，只要將下列指令加入作爲注釋，您都可以指定您現行的檔案要使用哪一辭典。 Ispell 開始拼字檢查時即可應用所指定的辭典：

```
<!-- Local IspellDict: english -->
<!-- Local IspellPersDict: ~/emacs/.ispell-english -->
```

假如您在 `.emacs` 檔案裡已經設定了預設辭典爲法文，那您可以將上述指令加入您所寫的英文文件的結尾，以便使用英文辭典。

5.3 文件的拼字檢查

若要檢查整份文件，可在文件內任一處使用 `Meta-x ispell-buffer` 指令。您也可以檢查文件某些部分的拼字：

- 將要檢查拼字的部分之開頭用 `Ctrl-Spc` 標註 (mark) (即所謂的 mark-set-command)，
- 然後標註到那部分的結尾，
- 再鍵入指令 `Meta-x ispell-region`.

Emacs 就會開始執行 Ispell。當遇到辭典裡所沒有的字時，程式會顯示這個字（通常以反白顯示），並要求您輸入下列指令：

- **spc** accepts the word, this time only,
- **i** accepts the word and inserts it in your personal dictionary,
- **a** accepts the word for this session,
- **A** accepts the word for this file, and inserts it in the local file dictionary
- **r** allows you to correct the word by hand
- **R** allows you to correct all the occurrences of the misspelled word,
- **x** stops the checking, and puts the cursor back in place,
- **X** stops the checking and leaves the cursor where it is, letting you correct your file; you will be able to continue the spell-checking later if you type `Meta-x ispell-continue`,
- **?** gives you online help.

如果 Ispell 找到類似的詞彙，它會將這些文字顯現在一小視窗內，每一個文字前都有號碼標明，只要鍵入您要的那個字彙的號碼， Ispell 就會以您所選的字取代錯字。

5.4 個人辭典與個別檔案辭典 (local file dictionary) 的比較

i 鍵讓您可以在您的個人辭典內插入一個新字，而 **A** 鍵讓您在個別檔案辭典內插入新字。

個別檔案辭典其實是一檔案結尾所插入一系列的字或評論， Ispell 每次檢查此檔案的拼字時都會重新讀取這個辭典。如此，您就可以鍵入一些只在這特定檔案能接受的字，而其他檔案則可不受此辭典影響。

在我看來，我認爲個人辭典最好是用來儲存一些主辭典所沒有的字，尤其是那些屬於各國語言的字（像是有連字符號 hyphenated 的字），及一些通用的特定名詞（像是 *Linux*），但在個人辭典內加入太多像是姓名的單字也不太好，因爲這些字就會被當作一般字使用了（您可以想像如果您把 *Tof* 加入個人辭典內， Ispell 可能就會在下列的例子裡，忽略這個錯字：[2]）

[2] *'When the going gets tof, the tof get going.' Tof* 是法文 *Christophe* 名字的縮寫。

5.5 鍵入時的拼字檢查

當您在鍵入文字時，只要您在這 **ispell-minor-mode** 模式下， Ispell 就會自動幫您檢查拼字。要啟動或關閉這模式，請鍵入這指令 `Meta-x ispell-minor-mode`。每當有 Ispell 查不到的字時，它就會嗶一聲來提醒您。

假如您覺得嗶一聲很煩人，只要鍵入指令 `Meta-x set-variable RET visible-bell RET t RET`，您就可以改用螢幕的閃爍來提醒您。您也可以在 `.emacs` 檔案裡加入下列指令來關掉 Emacs 的聲音：

```
(setq visible-bell t)
```

6 一些投機的技巧

6.1 自動插入頁首 (header)

在 Emacs 下，您可以將任何事件外掛 *(hook)* 任何的動作（像是打開一個檔案，儲存，或執行一個新的模式，等等）。

您也可以利用 **autoinsert** 程式庫來執行一些特別的動作：當您在 Emacs 下開新檔案時，這個程式庫就會根據檔案的屬性來插入相符的標準頁首。

就我們的情形，這標準頁首就是我們說明文件型態 (LinuxDoc)，標題，作者，及日期的地方。

接下來我會描述兩個如何插入這種頁首的方法。您可以插入一個含有相關資訊的模板 (template) 檔案，或者您也可以跑一跑 **elisp** 程式。

6.1.1 藉由插入一個檔案

當您開啓一份檔案時，首先您必須告訴 Emacs 執行自動插入，然後再讀取 **autoinsert** 程式庫的資料來宣告我們要改變的 `auto-insert-alist` 表單。這表單定義了不同檔案型態所要插入的頁首。它的預設值是要插入的檔案必須在 `~/insert/` 目錄下，但如果您要重新定義其 `auto-insert-directory` 也是可行的。

每一次您開啓一份新的 SGML 檔案時，只要在 `.emacs` 檔案內加入下列指示，您就可以插入 `~/emacs/sgml-insert.sgml` 檔：

```
(add-hook 'find-file-hooks 'auto-insert)
(load-library "autoinsert")
(setq auto-insert-directory "~/emacs/")
(setq auto-insert-alist
      (append '((sgml-mode .  "sgml-insert.sgml"))
              auto-insert-alist))
```

那您就可以在 `~/emacs/sgml-insert.sgml` 檔寫入您自訂的頁首，然後再重新執行 Emacs 來開啓某一 `foobar.sgml` 檔案，之後 Emacs 會詢問您確認這自動插入的指令來插入您要的頁首。

6.1.2 藉由執行一個程式

這方法跟上述大同小異，但不同的是您必須設定 auto-insert-alist 表單，告訴它要插入的是一個執行程式，而非一個檔案。只要在 .emacs 檔案內加入下列指示，就可以自動執行，不過當然您得先把這程式寫在一個叫做 ~/emacs/sgml-header.el 的檔案裡才可以。（鑒於這函式可能會過長，您不需要將它放在 .emacs 檔案裡）：

```
(add-hook 'find-file-hooks 'auto-insert)
(load-library "autoinsert")
(add-to-list 'load-path "~/emacs")
(load-library "sgml-header")
(setq auto-insert-alist
      (append '(((sgml-mode .  "SGML Mode") . insert-sgml-header))
              auto-insert-alist))
```

您可以參考A (附錄)裡有關 Insert-sgml-header 程式的例子。

A Insert-sgml-header程式之範例

讀者在撰寫一份 Linux Documentation Project 文件時，可以使用下列程式來啓動自動插入自訂的頁首。

透過一 *mini-buffer*，上述程式會要求讀者提供一些或許必要的資訊。

它會先詢問要輸出的標題是什麼，若您沒有提供，則頁首就不會顯現標題。再來是日期，作者，作者的電子郵件帳號與 home page （最後兩項可隨意填入）。

接著它會詢問翻譯者的名字。若無，請直接鍵入 *Return*。若有，程式就會接著詢問譯者的電子郵件帳號及 home page （同樣也是隨意填入）。

再來它會把您的輸入存入 current buffer，包括任何您輸入的資訊格式，及摘要和第一章的標籤。之後它會將游標導向您要鍵入摘要的的地方。

```
(defun insert-sgml-header ()
  "Inserts the header for a LinuxDoc document"
  (interactive)
  (let (title author email home translator email-translator home-translator date
              starting-point)
    (setq title (read-from-minibuffer "Title: "))
    (if (> (length title) 0)
        (progn
          (setq date (read-from-minibuffer "Date: ")
                author (read-from-minibuffer "Author: ")
                email (read-from-minibuffer "Author e-mail: ")
                home (read-from-minibuffer "Author home page: http://")
                translator (read-from-minibuffer "Translator: "))
          (insert "<!doctype linuxdoc system>\n<article>\n<title>")
          (insert title)
```

```
(insert "</title>\n<author>\nAuthor: ") (insert author) (insert "<newline>\n")
(if (> (length email) 0)
    (progn
      (insert "<htmlurl url=\"mailto:")
      (insert email) (insert "\" name=\"") (insert email)
      (insert "\"><newline>\n")))
(if (> (length home) 0)
    (progn
      (insert "<htmlurl url=\"http://")
      (insert home) (insert "\" name=\"") (insert home)
      (insert "\">\n<newline>")))
(if (> (length translator) 0)
    (progn
      (setq email-translator (read-from-minibuffer "Translator e-mail: ")
            home-translator (read-from-minibuffer "Translator home page: http://"))
      (insert "Translator : ")
      (insert translator)
      (insert "<newline>\n")
      (if (> (length email-translator) 0)
          (progn
            (insert "<htmlurl url=\"mailto:")
            (insert email-translator) (insert "\" name=\"")
            (insert email-translator)
            (insert "\"><newline>\n")))
      (if (> (length home-translator) 0)
          (progn
            (insert "<htmlurl url=\"http://")
            (insert home-translator) (insert "\" name=\"")
            (insert home-translator)
            (insert "\"><newline>\n")))))
(insert "</author>\n<date>\n")
(insert date)
(insert "\n</date>\n\n<abstract>\n")
(setq point-beginning (point))
(insert "\n</abstract>\n<toc>\n\n<sect>\n<p>\n\n\n</sect>\n\n</article>\n")
(goto-char point-beginning)
))))
```