LinuxDoc+Emacs+Ispell-HOWTO

di Philippe Martin (*feloy@wanadoo.fr*)
27 Febbraio 1998

Questo documento è dedicato a tutti coloro che scrivono o traducono gli HOWTO di Linux o qualunque altro documento per il Linux Documentation Project (anche ILDP, ovviamente!). Può fornire loro suggerimenti su come usare i vari applicativi tra cui Emacs e Ispell. Traduzione a cura di Christopher Gabriel, *cgabriel@pluto.linux.it* , ultima revisione 12 Ottobre 1998.

Indice

1 Presentazione

1.1 Copyright

1.2 Ringraziamenti

Ringrazio vivamente Sébastien Blondeel, che mi ha chiesto molto sulla configurazione di Emacs. Le sue intelligenti domande mi hanno permesso di capire meglio certe cose e di renderle disponibili a voi attraverso questo documento.

1.3 Commenti

Non esitate a dirmi qualunque cosa che pensate possa rendere questo documento migliore. Esaminerò le vostre critiche attentamente. Non esitate neppure a pormi domande riguardanti qualunque argomento trattato in questo documento. Sarò molto felice di rispondervi, visto che molto probabilmente questo mi permetterà di migliorare il mio lavoro [1].

1.4 Versioni

Questo documento prenderà in esame le seguenti versioni:

- Sgml-tools versione 0.99,
- Emacs versione 19.34,

[1]Nota del traduttore: Se questo italiano è orribile, allora mandatele a me!

- Ispell versione 3.1,
- Tutte le librerie di Emacs a cui fa riferimento questo documento sono distribuite con la versione di Emacs di cui sopra, ad eccetto di `iso-sgml`, che è fornito assieme a XEmacs, e `psgml`, che è una libreria a se stante.

2 Introduzione

2.1 SGML

Standard Generalised Mark-up Language, o **SGML**, è un linguaggio per definire la tipologia di un documento.

Per esempio, potete definire il tipo di documento *ricetta*, con una prima parte per la presentazione degli ingredienti, una seconda che introduce gli accessori, una terza che descrive le istruzioni passo passo per cuocere la torta, e una immagine finale che mostra il risultato, la torta pronta da mangiare.

Questo è chiamato *Document Type Definition (DTD)*, Definizione di Tipo di Documento. Non servirà a descrivere l'aspetto del prodotto finale, definisce solamente cosa può contenere.

Per usare nuovamente lo stesso esempio, sono sicuro che leggendo la mia idea di ricetta, riconoscerete le vostre ricette, o le vostre preferite. In ogni caso, attualmente hanno aspetto differente: le mie hanno una piccola immagine di una tazza nell'angolo in alto a sinistra, e gli ingredienti elencati nel giardino posteriore, fra la piscina e il barbecue. Le vostre?

Grazie a questa definizione standard, chiunque può scrivere un documento, senza doversi preoccupare di come apparirà al lettore.

2.2 La tipologia `LinuxDoc`

Questa tipologia è usata per scrivere, come potete avere indovinato, documenti relativi a Linux.

Questi documenti sono usualmente strutturati in questo modo: un titolo di inizio, seguito dal nome dell'autore, la versione del documento e la data. Successivamente è presente una breve descrizione (non avete quindi bisogno di girellare per tutto il documento per capire di cosa si parla o per realizzare che quello che c'è scritto non vi interessa), poi l'indice che mostra la struttura di tutto il documento, che vi permette di accedere direttamente alla sezione che più vi interessa.

È presente la lista dei capitoli, delle sezioni dei paragrafi. Attraverso questi, si può inserire piccole parti di programmi, cambiare i font per enfatizzare parole o frasi, inserire liste, fare riferimenti ad altre parti del documento etc.

Per scrivere questo tipo di documento, dovete specificare nel punto giusto il titolo, l'autore, la data, la versione, e ancora i capitoli e le sezioni, definire quando deve essere visualizzata una lista di oggetti, quali sono i suoi elementi etc.

2.3 `SGML-Tools`

Gli **SGML-Tools** permettono di creare dal documento con le vostre specifiche il risultato finale, nel formato che preferite. Se volete inserirlo nella vostra libreria privata, sceglierete *PostScript*. Se volete condividerlo attraverso il World Wide Web, sarà allora *HTML*. Se dovete leggerlo con Windows, potete trasformarlo in *RTF* in modo tale da renderlo leggibile con qualunque Word Processor. Oppure potete utilizzare tutti e tre questi formati, cosí sarete pronti a qualunque evenienza.

Gli SGML-Tools sono disponibili via FTP anonimo a *ftp://ftp.lip6.fr/pub/sgml-tools/*

3 Il vostro primo documento

3.1 Da un documento di testo

Se desiderate convertire un documento di solo testo in SGML per crearlo in altri formati, questo fa al caso vostro:

1. Aggiungete le seguenti linee all'inizio del file:

   ```
   <!doctype linuxdoc system>
    <article>
     <title>Qui metteteci il titolo</title>
     <author>
      nome dell'autore, email dell'autore etc
     </author>
     <date>
      versione e data
     </date>
   ```

2. Se volete aggiungere una breve descrizione del documento circondate il paragrafo con i tag `<abstract>` e `</abstract>`.

3. Dopodiché inserite il tag `<toc>`, che significa *Table of Contents*, ovvero l'*Indice*.

4. All'inizio di ogni nuovo capitolo, sostituite la linea contenente il numero e il titolo del capitolo con:

   ```
   <sect>Titolo del nuovo Capitolo
   ```

   e aggiungete il tag `</sect>` all fine del capitolo stesso.

   Nota : Non dovete inserire il numero del capitolo, questo viene inserito automaticamente.

5. Procede allo stesso modo per le sezioni. Dovete cancellare i loro numeri e circondate il titolo con il tag `<sect1>` e terminate con `</sect1>`.

6. Potete definite fino a 4 livelli di sezione nidificati, usando `<sectn>` ande `</sectn>` dove `n=` 2, 3, o 4.

7. All'inizio di ogni paragrafo, inserite il tag `<p>`.

8. Se dovete enfatizzare alcune parti del testo, usate il tag `<it>` e `</it>` (*italico*), `<bf>` e `</bf>` (**grassetto**), o `<tt>` e `</tt>` (`macchina da scrivere`).

9. Per inserire una lista come la seguente:

   ```
   Questa è una lista di quattro elementi:

    - qui va la prima linea
    - seguita dalla seconda
    - ancora un'altra, per favore
    - ok, basta così.
   ```

dovete sostituirla con:

```
 Questa è una lista di quattro elementi:
<itemize>
<item>qui va la prima linea
<item>seguita dalla seconda
<item>ancora un'altra, per favore
<item> ok, basta così.
</itemize>
```

10. Quando un intero blocco è parte di codice di un programma, e qualunque altra cosa deve essere evidenziata, tipo:

```
<verb>
 10 REM Oh mio Dio cosa è questo?
 20 REM Penso che ormai sia scomparso!
 30 PRINT "Sono tornato baby";
 40 PRINT "per salvare il mondo."
 50 INPUT "Da chi? chi sei tu? ",M$
 60 IF M$="Bill" THEN PRINT "Verso la saggezza.":GOTO PARADISE
 70 ELSE PRINT "Non hai via di scampo...":GOTO RICHMOND
 </verb>
```

11. In questo modo, il vostro documento SGML formattato avrà un aspetto decente. Se lo volete rifinire ulteriormenete, potete leggervi il manuale utente degli **SGML-Tools**, che fornisco ulteriori dettagli sul tipo di documento **LinuxDoc**.

4 Configurare Emacs

4.1 Caratteri accentati

Se volete scrivere documenti in francese o in qualunque altra lingua dell'Europa occidentale, avete bisogno del set di caratteri a 8-bit. Di seguito come configurare Emacs per fargli accettare questi caratteri.

4.1.1 La visualizzazione di caratteri 8-bit

Per permetter a Emacs di visualizzare caratteri a 8-bit, dovete aggiungere le seguenti linee nel vostro file `.emacs`:

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

Se state usando Emacs su di un terminale che non ha il supporto per i caratteri 8-bit, potete usare la libreria `iso-ascii` (`(load-library iso-ascii)`), che permette ad Emacs di visualizzare questo tipo di caratteri con la migliore approssimazione.

4.1.2 Digitare caratteri 8-bit

Se la vostra tastiera vi permette di digitare caratteri accentati, allora nessun problema. Se invece non è così ecco alcuni rimedi:

La libreria `iso-acc` La libreria `iso-acc` di Emacs vi permette di digitare caratteri 8-bit da una tastiera a 7-bit.

Per utilizzarla, inserite il seguente codice nel vostro file `.emacs`:

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

Quindi, mentre state usando Emacs e aprendo il file che volete editare, digitate `Meta-x iso-accents-mode`.

Potete inserire la **é** della parola francese *café* digitando **'** e `e`. Solitamente potete inserire una lettera accentata digitando prima l'accento, e poi la lettera da accentare (maiuscola o minuscola). I seguenti sono accenti che potete usare:

- **'** : Acuto
- **`** : Grave
- **^** : Circonflesso
- : Dieresi
- **~** : Tilde, cedilla, e altri casi particolari
- / : Per barrare una lettera etc.

Se avete bisogno di uno di questi caratteri e non della lettera accentata, inseritelo facendo seguire uno spazio. Per esempio, per inserire *l'éléphant*, premete `l '` ⟨*spc*⟩ `' e l ' e ...`

Potete trovare tutte le combinazioni possibili nel file `iso-acc.el`.

Il tasto ⟨*Meta*⟩ Alcuni terminali vi permettono di digitare caratteri a 8-bit con il tasto⟨*Meta*⟩ (oppure ⟨*Alt*⟩). Per esempio, premendo ⟨*Meta-i*⟩ otterrete il carattere **é**. Ma Emacs riserva al tasto ⟨*Meta*⟩ altri usi, e io so che non esiste nessuna libreria che permette di usarlo per i caratteri accentati.

Questa è una soluzione:

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

Una linea di questo genere, se inserita nel vostro `.emacs`, permetterà di digitare **é** usando la combinazione ⟨*Meta*⟩**-i**. Potete ridefinire in questo modo la combinazione che permetterà di inserire **i** nel tasto giusto e **351** nel giusto codice (il collegamento con il codice viene estratto dalla tabella del set di caratteri ISO-8859-1).

Attenzione! Qualche impostazione di localizzazione può ridefinire queste combinazioni.

4.1.3 La visualizzazione di caratteri SGML a 8-bit

Lavorando con SGML, potete accentare i caratteri con delle macro. Per esempio, il tasto **é** è **é**. Generalmente, le applicazioni che devono leggere SGML possono gestire caratteri a 8-bit e non c'è bisogno di usare queste macro. Ma alcune di queste non sono in grado di farlo. Ma, visto che c'è un modo per risolvere questo problema, sarebbe fastidioso vedere queste applicazioni non funzionare correttamente, no?

La libreria `iso-sgml` vi permette di digitare caratteri accentati sotto Emacs, come normale, ma quando il file sarà salvato su disco, allora questi caratteri saranno convertiti nel loro equivalente SGML.

È quindi molto semplice, grazie a questa libreria, digitare e rileggere il proprio documento con Emacs, e sarete sicuri che qualunque applicazione che non supporta i caratteri 8-bit potrà accettarlo.

Per usare questa libreria, dovete soltanto aggiungere le seguenti linee nel vostro file `.emacs`:

```
(setq sgml-mode-hook
'(lambda () "Default per SGML-mode"
 (load-library "iso-sgml")))
```

4.2 Modalità SGML

Una volta caricato un file che ha una estensione di tipo **.sgml**, Emacs entra automaticamente in modalità SGML, o **sgml-mode**. Se non è così, potete sempre avviarla manualmente digitando `Meta-x sgml-mode`, oppure automaticamente aggiungendo queste righe nel vostro `.emacs`:

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

Questa modalità vi permette di scegliere come inserire i caratteri a 8-bit per esempi. Con `Meta-x sgml-nome-8bit-mode` (o la voce di menu *SGML/Toggle 8-bit insertion*), potete scegliere di digitare i caratteri a 8-bit così come sono, oppure in forma SGML, per esempio **&...;**. Permette inoltre di nascondere o mostare i tag SGML, con `Meta-x sgml-hide-tags` e `Meta-x sgml-show-tags`.

4.3 Modalità PSGML

La modalità PSGML aiuta moltissimo a editare documenti SGML con Emacs.

La documentazione *psgml-linuxdoc*

spiega come installare questa modalità e usarla con *LinuxDoc*.

4.4 Miscellanea

4.4.1 Modalità auto-fill

In modalità normale, quando digitate un paragrafo e arrivate alla fine della linea, dovete usare il tasto ⟨*Invio*⟩ da soli per poter raggiungere la linea successiva, oppure la linea continuerà per tutto il paragrafo.

Se proseguite con alcune linee oltre una larghezza ragionevole, non sarete in grado di vedere il resto della linea con alcuni editors.

La modalità **auto-fill** automatizza questa procedura, peraltro noiosa: quando arrivate a una determinata colonna (solitamente la settantesima), sarete automaticamente spostati nelle linea successiva.

Il testo seguente descrive come utilizzare questa modalità, e impostare la larghezza delle vostre linee a 80:

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

5 Ispell

Se volete controllare l'ortografia dei vostri documenti all'interno di Emacs, allora potete usare il pacchetto **Ispell** e la sua modalità Emacs.

5.1 Impostare il dizionario predefinito

Potete impostare Emacs in modo tale che quando un file viene caricato scelga automaticamente quale dizionario usare (potete selezionarne diversi). Il primo, sicuramente il piú importante, è il dizionario distribuito con Ispell. Potete scegliere fra varie lingue. Il secondo è il vostro dizionario personale, dove Ispell inserirà le parole che non può trovare nel dizionario principale ma che devono essere ricordate. Se desiderate usare come dizionario predefinito quello Francese che è distribuito con Ispell, e se volete usare il file `.ispell-dico-perso`, che si trova nella vostra directory home, come dizionario personale, inserite le seguenti linee nel vostro file `.emacs`:

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

5.2 Selezionare un dizionario per determinati file

Potreste avere qualche problema se non si fate la correzione ortografica dei vostri documenti sempre con la lingua. Se traducete documenti, è normale che passiate da un linguaggio (e dizionario) all'altro molto spesso.

Non sono a conoscenza di una funzione Lisp che permetta di selezionare, sia automaticamente oppure con un click del mouse, il dizionario principale e quello personale associato al linguaggio che viene utilizzato (se qualcuno lo sa, per favore me lo dica!).

In ogni caso, è possibile indicare, alla fine del file, quale dizionario volete che venga utilizzato per quel determinato documento. È sufficiente aggiungere queste specifiche come commenti, in modo tale che Ispell possa leggerli quando viene avviata la correzione ortografica. Ecco un esempio per la lingua inglese:

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

Se avete precedentemente definito, nel vostro file `.emacs`, che il vostro dizionario è, per esempio, quello francese, allora potete aggiungere queste linee alla fine di ogni file scritto in inglese.

5.3 Correzione ortografica dei vostri documenti

Per eseguire la correzione ortografica del vostro documento, usate, in qualcunque punto del documento, il comando `Meta-x ispell-buffer`. Potete in ogni caso eseguire questo controllo in una singola sezione del documento:

- Marcare l'inizio della sezione con `Ctrl-Spc` (mark-set-command),
- Muovetevi fino alla fine della sezione da correggere,
- digitate `Meta-x ispell-region`.

A questo punto Emacs avvia Ispell. Appena incontra una parola sconosciuta, questa viene mostrata evidenziata e aspetta che immettiate un comando:

- **spazio** accetta la parola, questa volta soltanto
- **i** accetta la parola e la inserisce nel dizionario personale
- **a** accetta la parola per questa sessione di Ispell
- **A** accetta la parola per questo file, e la inserisce nel dizionario personale
- **r** permette di correggere la parola manualmente
- **R** permette di correggere tutte le ricorrenze della parola che è stata corretta
- **x** arresta la correzione, e riporta il cursore dove si era lasciato inizialmente
- **X** arresta la correzione e lascia il cursore dove si trova, permettendo di correggere il file; sarete in grado di continuare la correzione in un secondo tempo usando `Meta-x ispell-continue`
- **?** accede alla guida in linea.

Se Ispell trova una o più parole nel dizionario simili a quella sconosciuta, le mostrerà in una piccola finestra, ognuna preceduta da un numero. Basta premere questo numero per sostituire la parola sbagliata con quella corrispondente.

5.4 Dizionario personale VS dizionario nel file locale

Il tasto **i** vi permette di inserire una parola nel vostro dizionario personale, come **A** permette di inserirla nel dizionario in file locale.

Il dizionario nel file locale è una sequenza di parole inserite alla fine del file, come commento, rilette da Ispell ogni volta che viene eseguito sul quel file specifico. In questo modo è possibile accetare alcune parole nel file in questione, ma non in altri.

Dal mio punto di vista credo che sia meglio che il dizionario personale venga utilizzato per parole che non sono contenute nel dizionario principale ma che sono tipiche del linguaggio (come le parole accentate), con in piú parole comuni come nomi propri o altre (come *Linux*), se non sono troppo simili ad altre presenti in quello principale; aggiungere troppe parole nel dizionario personale, come nomi propri, può essere pericoloso, poiché possono essere viste come parole della lingua in uso (immaginate Ispell alle prese con qualcosa del genere: '*When the going gets tof, the tof get going*').[2]

[2] *Tof* è l'abbreviazione francese per il nome proprio *Christophe*!.

5.5 Correzione ortografica durante la digitazione

Ispell può controllare l'ortografia mentre state digitando il vostro documento. Dovete utilizzare **ispell-minor-mode** per questo. Per avviarlo o arrestarlo, digitate `Meta-x ispell-minor-mode`. Ispell emetterà un *beep* ogni volta che digiterete una parola che non è nei due dizionari.

Se questi *beep* vi stanno annoiando (oppure il vostro compagno di stanza si è stancato di voi..), potete rimpiazzarli con un flash sullo schermo, con il comando `Meta-x set-variable RET visible-bell RET t RET`. Potete anche aggiungere le seguenti linee nel vostro file `.emacs` e Emacs non parlerà mai piú:

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

6 Trucchetti Sporchi

6.1 Inserire automaticamente un header

Emacs permette di associare delle procedure ad eventi specifici (aprire un file, salvarlo, avviare una nuova modalità etc).

La libreria **autoinsert** utilizza questa caratteristica: quando aprite un nuovo file con Emacs, questa libreria inserisce, dipendentemente dal tipo di file, un header *standard*.

Nel nostro caso, questo header *standard* può definire il tipo di documento (LinuxDoc), il titolo, l'autore, e la data.

Descriverò adesso due modi per fare questo. Potete creare un template che contenga le informazioni da inserire, oppure potete avviare una routine **elisp**.

6.1.1 Inserendo un file

Per prima cosa dovete permettere a Emacs di avviare la `auto-insert` quando un file viene aperto, quindi di leggere la libreria `auto-insert` che dichiara la `auto-insert-alist` che necessita di cambiamenti. Questa lista definisce gli header da inserire per ogni tipo di file. Per default, il file da inserire deve trovarsi nella cartella `~/insert/`, ma è possibile definire la variabile `auto-insert-directory` se volete mettere i vostri file da qualche altra parte.

Aggiungere le seguenti linee al vostro `.emacs` per il inserire il file `~/emacs/sgml-insert.sgml` ogni volta che aprite un nuovo documento SGML:

```
(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))
```

A questo punto potete scrivere nel file `~/emacs/sgml-insert.sgml` il vostro header personalizzato, quindi riavviare Emacs e aprire qualche nuovo file .sgml. Emacs dovrebbe chiedervi la conferma per l'inserimento automatico, e se rispondete 'Yes', inserire il nuovo header.

6.1.2 Avviando una routine

Questo funziona come prima, ma invece di impostare la `auto-insert-alist` in un file da inserire, dovete impostare una funzione da eseguire. Questo è il procedimento, assumendo come esempio che volete scrivere una funziona in un file chiamato `~/emacs/sgml-header.el` (non è il caso di appesantire il vostro `.emacs` con questo tipo di funzioni, oppure diventerà troppo grande):

```
(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))
```

Troverete nella 7 (appendice) un esempio di una funzione `insert-sgml-header`.

7 Una funzione insert-sgml-header

Questa funzione permette all'utente di inserire un header personalizzato in un documento per il Linux Documentation Project all'interno di un file. Può essere richiamata automaticamente quando viene aperto un nuovo file, oppure in modo esplicito dall'utente.

Questa funzione attende un input dell'utente attraverso il *mini-buffer*, per alcune informazioni, di cui alcune sono necessarie, altre invece no.

Per prima cosa il titolo. Se non viene inserito, la funziona esce immediatamente, e niente viene inserito. Segue poi la data, l'autore, la sua email e home-page (le ultime due sono opzionali).

Successivamente c'è una richiesta per il nome del traduttore. Se non volete inserire niente, basta premere *Invio*, e nessun altra domanda circa un ipotetico traduttore verrà richiesta. Se invece ne esiste uno, verrà richiesta la sua email e home-page (come sempre opzionali).

Questa funzione quindi inserirà nel file corrente tutte le informazioni da voi inserite, inclusi naturalmente tutti i tag necessari. Inserirà anche quelli per l'abstract e per il primo capitolo. Infine posizionerà il cursore dove l'abstract deve essere inserito.

```
(defun insert-sgml-header ()
  "Inserisce gli header per un documento LinuxDoc"
  (interactive)
  (let (title author email home translator email-translator home-translator date
             starting-point)
    (setq title (read-from-minibuffer "Titolo: "))
    (if (> (length title) 0)
        (progn
          (setq date (read-from-minibuffer "Data: ")
                author (read-from-minibuffer "Autore: ")
                email (read-from-minibuffer "E-mail autore: ")
                home (read-from-minibuffer "Home page autore: http://")
                translator (read-from-minibuffer "Traduttore: "))
          (insert "<!doctype linuxdoc system>\n<article>\n<title>")
```

```
(insert title)
(insert "</title>\n<author>\nAutore: ") (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 "E-mail traduttore: ")
            home-translator (read-from-minibuffer "Home page traduttore: http://"))
      (insert "Traduzione : ")
      (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)
))))
```