HOWTOs with LinuxDoc

David S. Lawyer v0.09, November 2007

Questo testo tratta di come scrivere gli HOWTO usando il semplice linguaggio a marcatori (markup) LinuxDoc. È rivolto principalmente agli autori del Linux Documentation Project (e ad autori futuri alle prime armi che vogliono iniziare rapidamente). Se si vuole usare DocBook, il linguaggio a marcatori più completo e difficile, (incluso XML), vedere la LDP Authoring Guide (Guida per gli autori di LDP). Traduzione ed adattamenti in italiano a cura di Beatrice Torracca, beatricet (at) libero (dot) it, e Hugh Hartmann, hhartmann (at) libero (dot) it, revisione a cura di Vieri Giugni, v.giugni (at) gmail (dot) com). Per versioni aggiornate di questo documento, e per trovare altra documentazione in italiano sul software libero, visitare il sito dell' *ILDP* http://it.ildp.org

Indice

1 Introduzione

1.1 Per partire immediatamente

Per imparare solamente LinuxDoc, si salti a 5 (Imparare LinuxDoc). Se si vuole iniziare a scrivere immediatamente, provare a completare gli spazi vuoti in questo modello, che genera output in formato LinuxDoc:

The LDP HOWTO Generator `http://www.nyx.net/~sgjoen/The_LDP_HOWTO_Generator.html` . Può essere usato per iniziare a scrivere il proprio HOWTO e lo si può finire in seguito usando un editor di testi sul proprio PC.

1.2 Copyright e licenza

1.3 Perché si dovrebbe scrivere un HOWTO?

Si conoscono cose su Linux per le quali non è disponibile nessuna buona documentazione libera e che potrebbero essere utili ad altri? Anche se non si conosce la materia approfonditamente, è sempre possibile scriverne se si ha la passione, la capacità e la volontà di imparare nuove cose su di essa e se si ha il tempo per farlo. Si è capaci di scrivere chiaramente usando un elaboratore o un editor di testi? Si vogliono aiutare migliaia di altre persone e permettere loro, senza alcun costo, di leggere ciò che si scrive? Una volta che il documento è stato scritto, si è disponibili a ricevere, via posta elettronica, suggerimenti dai lettori e ad usare selettivamente queste informazioni per migliorare il proprio HOWTO? Si è favorevoli ad avere il proprio documento disponibile su centinaia di siti web in tutto il mondo? Se si può rispondere Sì a queste domande, allora si è incoraggiati a scrivere qualcosa per il Linux Documentation Project (LDP); ma si sia preparati all'idea che ci potrebbe volere più tempo di quanto previsto.

1.4 Perché ho scritto questo documento

Perché ho scritto questo documento quando c'è già una LDP Authoring Guide (Guida per gli autori LDP)? Perchè la guida LDP è un lavoro lungo e dettagliato. Se si desidera iniziare rapidamente, è necessario qualcosa di molto più semplice e breve. Grazie a Matt Welsh per il suo file example.sgml che è stato usato come fonte principale di informazioni per le sezioni di esempio.

2 Informazioni sulla scrittura di un HOWTO

2.1 Copyright

Il copyright di tutti gli HOWTO e degli altri documenti LDP è dei rispettivi autori cosicché l'LDP non possiede diritti speciali sui vostri scritti. Accetta soltanto quei documenti che hanno una licenza che permetta a chiunque di copiarli e distribuirli. Incoraggia gli autori ad usare una licenza che permetta anche modifiche; in questo modo, se l'autore smette di mantenere un documento, qualcun altro può farlo. Per maggiori dettagli si guardi il Manifesto di LDP.

2.2 Scegliere un argomento

Se non si è sicuri dell'argomento su cui scrivere, guardare alcuni dei documenti dell'LDP, inclusi quelli in

HOWTO non mantenuti `http://www.tldp.org/authors/unmaint.html` . Si scelga un argomento per cui si prova interesse e che necessita di una buona documentazione. Se si trova qualcosa di già scritto e attualmente mantenuto che ha bisogno di essere migliorato, si cerchi per prima cosa di contattare l'autore e di sottoporre suggerimenti. Se non si riesce a contattare l'autore, si guardi la licenza per vedere se si è autorizzati a modificare il documento. Anche nei casi in cui non sia possibile fare modifiche o migliorare il documento, si può comunque scrivere un nuovo documento sullo stesso argomento partendo da zero, usando un'altra impostazione e nuove fonti di informazione.

3 Il formato degli HOWTO

3.1 Introduzione

Gli HOWTO dell'LDP sono rilasciati al pubblico in vari formati: testo semplice, HTML, PostScript e PDF. Invece di dover scrivere lo stesso HOWTO in tutti questi formati, si scrive solamente un HOWTO in un formato sorgente, DocBook o LinuxDoc, che viene convertito dal computer in tutti gli altri.

Per avere un'idea di come appaia questo formato basta guardare il file sorgente di una pagina web (se non lo si è già fatto). Si vedranno tante parole racchiuse tra due <parentesi angolari>: queste sono chiamate tag. Queste pagine web (tag e tutto il resto) sono in html: *Hypertext Markup Language*. L'LDP usa qualcosa di simile per i propri documenti.

I linguaggi a marcatori che usa l'LDP soddisfano o i requisiti dello *Standard Generalized Markup Language (SGML)* o quelli di *XML*. Attualmente LDP usa le due seguenti varianti di sgml: *LinuxDoc* e *DocBook*, oltre alla variante *DocBook* di XML. È interessante notare che l'html nasce come un'altra variante di sgml (ma molte funzioni che vengono usate nell'html violano le regole dell'sgml, perciò non è più sgml puro).

Questo mini-HOWTO è totalmente orientato all'utilizzo di LinuxDoc, una variante semplice di sgml. Si può chiamarla linguaggio a marcatori LinuxDoc. Può essere convertita automaticamente dal computer in html, testo semplice, postscript, pdf e DocBook. È molto più semplice rispetto all'HTML o al DocBook e non è necessario un editor speciale in quanto è facile scrivere i tag (o usare macro per essi) nel proprio editor o elaboratore di testi preferito.

4 LinuxDoc e DocBook a confronto

Prima di leggere questa sezione è bene avere almeno una conoscenza di base dei tag nei linguaggi di marcatura. Perciò se non si sa bene cosa siano, può essere una buona idea guardare l'5.2 (Esempio 1) per i marcatori di LinuxDoc.

Un modo per fare un paragone è quello di analizzare HOWTO reali sul sito dell'LDP; per trovarli, fare clic su

Indice dei DocBook `http://cvs.tldp.org/go.to/LDP/LDP/howto/docbook/` oppure su

Indice dei LinuxDoc `http://cvs.tldp.org/go.to/LDP/LDP/howto/linuxdoc/` .

Si noterà che i documenti DocBook sono pieni zeppi di tag mentre quelli LinuxDoc hanno meno tag e sono perciò più leggibili e più facili da scrivere e modificare. Alcuni documenti sul sito LDP hanno marcatori più chiari di altri, perciò sarebbe bene guardarne più di uno.

Si può inizialmente pensare che DocBook sia più avanzato poiché in esso ci sono così tanti tag in più, ma ciò non è necessariamente vero. Se si crea un documento in LinuxDoc e lo si converte in formato DocBook (con il computer) ci saranno più tag nella versione DocBook, inclusi tag nuovi che non erano presenti nella versione LinuxDoc. Perché? Perché LinuxDoc permette di omettere alcuni tag, mentre DocBook no. In questo senso LinuxDoc è più flessibile e avanzato di DocBook; non solo permette spesso di omettere i tag di chiusura come </title>, ma permette di omettere coppie complete di tag iniziali e finali. In LinuxDoc, per esempio, i paragrafi sono separati normalmente da righe vuote invece che dalla coppia di tag <p> (<para> in DocBook) e in questo modo i tag di paragrafo raramente sono necessari. Un altro esempio è l'inizio di una nuova sezione in un documento LinuxDoc; è sufficiente scrivere il titolo della sezione dopo il tag <sect> mentre DocBook richiede che si racchiuda il titolo in una coppia di tag <title>.

Quando si esegue un programma per convertire LinuxDoc in HTML, per esempio, la prima cosa che il programma fa è di cercare tutti i tag omessi e aggiungerli al documento. Viene aggiunta, per esempio, una coppia di tag equivalenti alla coppia <title> dopo un tag <sect> (inclusi i tag <sect1>, ecc.). Perciò, in un certo senso, anche LinuxDoc ha molti tag, ma sono nascosti all'utente per rendere il documento molto più facile sia da scrivere sia da leggere. Nella maggior parte dei casi, l'autore di un documento in LinuxDoc non è a conoscenza dell'esistenza della maggior parte dei tag mancanti. Non è necessario (normalmente) che l'autore li conosca, dato che esistono solo nella memoria del computer (o in un file temporaneo) quando il PC fa la conversione da LinuxDoc a un qualche altro formato (come HTML). In realtà è possibile salvare un file che mostri i tag aggiunti, ma ciò viene fatto per lo più da programmatori durante il debug di codice relativo a LinuxDoc (o da persone come me che sono curiose di sapere come funzioni).

I tag DocBook sono spesso più lunghi dei loro equivalenti LinuxDoc come <emphasis> rispetto a <em>. Per un breve confronto si veda il sito web dell'autore:

Comparison of DocBook to LinuxDoc `http://www.lafn.org/~dave/linux/ld_vs_db.txt` (Confronto tra DocBook e LinuxDoc).

Il fatto che molti tag siano omessi durante la scrittura in LinuxDoc (ma vengano recuperati dal software) è tuttavia una sola delle ragioni per cui i documenti LinuxDoc hanno meno tag. Un'altra ragione è che DocBook ha in realtà molti più tag da usare di quanti non ne abbia LinuxDoc. Quest'ultimo per esempio ha un unico tag per il nome dell'autore, mentre DocBook ha tag separati per il nome, il secondo nome e il cognome. Perciò in questo senso DocBook è più potente di LinuxDoc, ma di converso è più debole perché non può recuperare i tag mancanti. La soluzione a questa dicotomia sarebbe quella di unire LinuxDoc e DocBook così da mantenere i vantaggi di entrambi. DocBook dovrebbe allora abbandonare il suo attuale stato di linguaggio XML poiché quest'ultimo vieta l'omissione di tag ed altro, mentre SGML lo permette.

Uno dei motivi per cui non è permesso omettere tag in DocBook è che ciò rende più facile per i programmatori la scrittura di software che lo analizzi e lo converta in altri formati. Non è necessario che il software sia capace di individuare e aggiungere i tag mancanti. Fare in modo che le cose siano più semplici per i programmatori, le rende però più difficili per il molto più vasto numero di scrittori che devono usare DocBook.

Ci sono editor o elaboratori di testo speciali, come Lyx e Bluefish, che rendono più facile scrivere documenti DocBook. Bluefish, per esempio, aggiunge automaticamente i tag di chiusura. Per coloro che non vogliono imparare ad usare un nuovo editor o elaboratore di testo, LinuxDoc è però molto più facile dato che si possono inserire i tag a mano o creare un insieme di macro per inserire i tag. Io uso l'editor vim e se digito ;s esso inserisce il tag <sect1>; ;r inserisce <sect>, ;i inserisce <item>, eccetera. Per i tag delle intestazioni nelle prime righe di un documento, basta copiarle da un altro documento e cambiare le parole dopo i tag. Naturalmente io non devo modificare il nome o l'indirizzo dell'autore. Un grande vantaggio di LinuxDoc è perciò di poterlo usare facilmente con lo stesso editor o elaboratore di testi normalmente utilizzato e con cui si ha familiarità.

Alcune persone che non capiscono la situazione hanno sostenuto l'idea che DocBook può essere semplice come LinuxDoc semplicemente usando un sottoinsieme dei tag DocBook. Ciò non funziona perché con DocBook continua ad essere necessario un numero di tag diverse volte più grande di quello necessario per ottenere lo stesso risultato in LinuxDoc a causa del fatto che DocBook non permette l'omissione di tag.

Nonostante i vantaggi di LinuxDoc, il numero di persone che usa DocBook è molto più grande di quello delle persone che usano LinuxDoc, in parte perché DocBook è stato promosso dalle case editrici. Esiste un programma di Reuben Thomas (ld2db) che può convertire documenti LinuxDoc in DocBook. Non è perfetto e molto probabilmente sarà necessario fare modifiche manuali. Inoltre l'LDP converte automaticamente un HOWTO scritto in LinuxDoc in formato DocBook, dopo che questo è stato consegnato. Sarebbe bello se esistesse un altro programma che convertisse automaticamente i documenti DocBook in LinuxDoc in modo da permettere la loro modifica usando il più semplice linguaggio a marcatori LinuxDoc.

5 Imparare LinuxDoc

5.1 Introduzione

LinuxDoc è molto più facile da imparare rispetto a DocBook. Molto di ciò che si impara riguardo LinuxDoc, tuttavia, sarà utile anche per DocBook. Così, se eventualmente si decidesse di usare DocBook, la maggior parte dello sforzo speso per l'apprendimento di LinuxDoc non sarà sprecato.

Un modo per imparare LinuxDoc è attraverso degli esempi. Ho scritto tre file d'esempio di difficoltà che va da facile a intermedia. Il contenuto di questi file è stato copiato in questo HOWTO. Per trasformarli in file individuali si può tagliarli dal testo (partendo dal primo tag) e scriverli in un file. Poi si può tentare di trasformarne uno in un testo semplice usando ad esempio sgml2txt –pass=-P-cbou un-esempio.sgml per vedere come appare. Assicurarsi che i nomi dei file finiscano in .sgml.

Se si desidera vedere alcuni esempi reali si può andare su un sito mirror di LDP, trovare gli HOWTO e selezionare LinuxDoc SGML. Oppure si può andare direttamente al sito principale:

Indice degli Howto (LinuxDoc) http://cvs.tldp.org/go.to/LDP/LDP/howto/linuxdoc/ .

Ora il primo semplice esempio.

5.2 Esempio 1 (nome file: esempio1.sgml)

```
<!doctype linuxdoc system>
<article>
<title>Primo esempio (esempio1)
<author>David S.Lawyer

<sect>Introduzione
<p>Questo rappresenta un esempio molto semplice di "sorgente" per il
   sistema di formattazione testi LinuxDoc. Questo paragrafo inizia con
   un tag di paragrafo (una "p" racchiusa tra due parentesi angolari).
   Notare che ci sono altri tag, anch'essi racchiusi tra parentesi
   angolari. Se non si vede nessun tag, allora si sta leggendo un file
   convertito, si cerchi invece il file sorgente: esempio1.sgml
   (che contiene i tag).

   Questo è il paragrafo successivo. Si noti che solo una riga
   vuota lo separa dal paragrafo precedente.
   Pertanto non  è necessario mettere un tag "p" al suo inizio.
   Il tag "p" è necessario solamente per il primo paragrafo di
```

```
   una sezione (appena dopo il tag di sezione). Il suffisso sgml del
   file significa Standard Generalized Markup Language. Quella che si
   sta leggendo  è la variante LinuxDoc di sgml, come specificato
   nella primissima riga di questo file.

<sect>I tag
<p>I tag sono parole all'interno di parentesi angolari. Il tag "sect"
   qui sopra marca l'inizio di una nuova sezione di questo documento di
   esempio. La prima sezione era "Introduzione" e ora si sta leggendo la
   seconda sezione intitolata "I tag". Se questo fosse un documento
   lungo (come ad esempio un libro), una sezione corrisponderebbe ad un
   capitolo.

   Si noti che, all'inizio di questo articolo, ci sono i tag "article"
   (articolo), "title" (titolo) e "author" (autore). Alla fine di questo
   articolo un tag "/article" ne segna la fine.
   C'è dunque una coppia di tag "article", il primo
   è il tag d'apertura e il secondo il tag di chiusura. Una coppia di
   tag "article" racchiude dunque l'intero articolo.
   Negli esempi successivi, si vedranno altri tag che
   vanno in coppia come questi. Essi hanno effetto su tutto quello
   che si trova tra la coppia (il tag iniziale e il tag finale). Ogni
   tag il cui nome sia immediatamente preceduto dal simbolo "/" è
   un tag di chiusura.

   Quando questo codice sorgente viene convertito in un altro formato
   (come in formato testo semplice usando il programma sgml2txt) i tag
   vengono rimossi. I tag aiutano solamente il programma sgml2txt a fare
   la conversione. Ci sono molti altri tag da imparare; quando questo
   primo esempio è stato compreso, si passi dunque all'esempio
   successivo, l'esempio 2. Non si devono realmente
   memorizzare i tag, dato che saranno ripetuti (ma con
   poca o nessuna spiegazione) negli esempi successivi.
</article>
```

5.3 Esempio 2 (nome file: esempio2.sgml)

```
<!-- Questo è un commento. Viene ignorato quando questo file
     sorgente viene convertito in altri formati.  -->
<!-- Il tag sotto a questo indica il formato di questo file:
     LinuxDoc -->
<!doctype linuxdoc system>

<article>

<title>Secondo esempio (esempio2)
<author>David S. Lawyer
<date>v1.0, July 2000

<abstract>
```

```
Questo paragrafo rappresenta il sommario. Questo documento è il secondo
esempio sull'utilizzo della variante LinuxDoc-SGML di sgml. È
più complesso rispetto al primo esempio (esempio1.sgml), ma
più semplice del terzo esempio (esempio3.sgml). Dopo averlo
assimilato, si sarà in grado di scrivere un semplice HOWTO
usando LinuxDoc. Fine del sommario.
</abstract>

<!-- "toc" = l'indice. Sarà creato in questo punto. -->
<toc>

<!-- La  parte principale dell'articolo (o del documento) inizia qui. La
     parte soprastante rappresenta una specie di lunga intestazione. -->

<sect>Questo secondo esempio (esempio2.sgml)
<p>A meno che non si abbia confidenza con i linguaggi a marcatori, si
   dovrebbe leggere prima l'esempio 1. Potrebbe essere utile dare in pasto
   questi file di esempio ad un "traduttore" come sgml2txt, per
   convertirli in testo e notare come il risultato appaia differente
   ripetto a questo documento "sorgente" con tutti i suoi tag.

<sect>Impaginazione dell'articolo
<sect1>Corpo del documento
<p>Dopo l'intestazione viene il corpo del documento, che consiste
   in sezioni annidate marcate dai tag di sezione ("sect"). Le
   sottosezioni sono marcate con i tag "sect1". Dato che questa
   è la prima sottosezione all'interno della seconda sezione
   principale, essa è la sezione 2.1. All'interno di una
   sottosezione marcata da "sect1" ci possono essere sotto-sottosezioni
   come "sect2". Ci sono persino tag come "sect3", "sect4", ecc., ma
   è improbabile che sia necessario usarli. Notare che i tag
   nell'uso reale vanno racchiusi tra parentesi angolari, < e >.

<sect2>Questa è una sotto-sottosezione
<p>
   Si tratta della sezione 2.1.1. Si noti che il tag "p" può essere
   su una riga da solo. Questo non cambia nulla nel documento
   risultante.

<sect1>Intestazione del documento
<p>Un modo per creare una parte di intestazione consiste nel copiarla da
   un altro file .sgml. Basta poi sostituire ogni cosa, eccetto i tag,
   con le informazioni corrette per il proprio documento. È come
   usare un modello.

<sect>Ulteriori caratteristiche nell'esempio 3
<p>Con i tag in questo esempio 2 si può scrivere un semplice
   breve documento lungo qualche pagina. Ma per documenti più
   lunghi o per altre importanti caratteristiche, come l'inserimento di
   collegamenti all'interno del documento, si deve studiare
   l'esempio successivo, l'esempio 3. Esso mostra anche come
```

```
  creare elenchi e usare tipi di carattere.
</article>
```

5.4 Esempio 3 (nome file: esempio3.sgml)

```
<!doctype linuxdoc system>
<!-- Notare il "mailto:" dopo il mio nome. Esso permette al
     lettore del formato HTML di cliccare sul mio indirizzo di posta
     elettronica per inviarmi un messaggio. -->

<article>
<title>Terzo esempio (esempio3)
<author>David S. Lawyer <url url="mailto:dave@lafn.org">
<date>v1.0, July 2000
<abstract>
Questo documento rappresenta il terzo esempio sull'uso della variante
LinuxDoc di sgml. È più complesso rispetto al secondo
esempio.
</abstract>
<!-- Commento: toc = Indice -->
<toc>

<sect>I tipi di carattere
<p>Sebbene non compaiano nell'output in testo semplice, funzionano per
   altre conversioni.
<bf>carattere grassetto</bf>             <em>carattere enfatizzato</em>
<sf>sans serif</sf>                      <sl>carattere inclinato</sl>
<tt>carattere a spaziatura fissa</tt>    <it>carattere corsivo</it>
Un altro modo per ottenere questi stessi tipi di carattere consiste nel
racchiudere il testo con il simbolo "/", in questo modo:
<bf/carattere grassetto/                 <em/carattere enfatizzato/
<sf/sans serif/                          <sl/carattere inclinato/
<tt/carattere a spaziatura fissa/        <it/carattere corsivo/
   Si noti che DocBook non ha tag per i tipi di carattere dunque
   sarebbe meglio non usarli se si ha in progetto di fare la conversione
   in DocBook.

<sect>I collegamenti <label id="links_">
<p>Si possono creare collegamenti (qualcosa su cui si può
   cliccare all'interno di un navigatore html per andare da qualche
   altra parte). Possono indirizzare semplicemente in un'altra parte di
   questo documento (riferimenti incrociati), tipo l'etichetta ("label")
   qui sopra, oppure possono indirizzare ad un sito web in Internet.

<sect1>Riferimenti incrociati
<p>Se si clicca su <ref id="links_" name="Collegamenti"> si
   viene spostati all'inizio della sezione precedente "I
   collegamenti" (che è etichettata come "links_"). L'etichetta
   può essere una qualsiasi parola liberamente scelta, ma
   è buona norma evitare parole comuni, in modo da poter
```

```
   cercare etichette univoche usando il proprio editor. Per questa
   ragione è stato usato links_ (con il simbolo di
   sottolineatura). Il nome di questo collegamento viene mostrato
   (nel formato html) come il nome su cui fare clic. Questo nome
   (Collegamenti) è presente anche nella versione in puro
   testo.

<sect1>Collegamenti con URL
<p>Se si clicca su <url url="http://www.tldp.org"> si viene spostati
   sul sito web del Linux Documentation Project. Il successivo
   collegamento aggiunge un nome su cui gli utenti cliccano:
   <url url="http://www.tldp.org" name="Linux Documentation Project">.
   Usando questo secondo metodo, non si deve nemmeno
   spiegare dove porta il collegamento, in quanto è ovvio dal
   nome.

<sect>Caratteri proibiti
<p>Ogni parola scritta tra parentesi angolari viene
   interpretata come un tag. Come fare allora per mostrare un tag in un
   documento? Per questo scopo si usa una parola in codice per i
   caratteri delle parentesi angolari.

   Si può usare &lt per il simbolo < e &gt per il simbolo
   >. lt = "less than" (minore di), gt = "greater than" (maggiore di).
   Per esempio, questo è un tag p: &lt;p&gt;.
   Naturalmente di fatto non inizia alcun paragrafo, ma appare
   nei documenti convertiti come <p>. Tutti questi codici iniziano con
   un carattere "&". Il carattere ";" posto dopo lt serve per
   separarlo. Non è necessario se c'è uno spazio dopo di
   esso. Ad esempio: 3 &lt 4. Di fatto, se si sapesse che va bene
   usare un > disaccoppiato si potrebbe scrivere <p> come &lt;p>.
   Questo non sarebbe riconosciuto erroneamente come tag, dato che
   non c'è l'apertura <. A dire il vero anche 3 < 4 funziona
   bene.

   Ci sono altri caratteri che non si possono mettere direttamente nel
   testo del documento. Per il carattere "&" in un comando AT del modem
   usare: AT&amp;. Se altri caratteri danno dei problemi (lo fanno
   raramente) si veda
   <ref id="ch_codes" name="Codici dei caratteri (macro)"> o la "guida"
   che viene fornita con linuxdoc-tools o sgml-tools.

<sect>Verbatim, codice e a capo
<sect1>Verbatim
<p>Se si vuole essere sicuri che il testo appaia esattamente come lo si
   è digitato, anche dopo la conversione in altri formati, usare
   il tag verbatim "verb". È utile per creare tabelle, e simili.
   Tuttavia alcune cose vengono comunque riconosciute come marcatori anche se
   si trovano tra tag verbatim. Tra queste sono incluse le macro che
   iniziano con il carattere "&" e i tag finali con il carattere "/".
```

```
<tscreen><verb>
% sgml2txt --pass="-P-cbou" esempio.sgml
</verb></tscreen>
Il tag "tscreen" imposta il carattere a spaziatura fissa e imposta il
rientro in modo bello da vedersi.

<sect1>Codice
<p>Questo racchiude del codice del
   computer tra due righe tratteggiate.
<tscreen><code>
Mettere il codice sorgente qui.
</code></tscreen>

<sect1>A capo
<p>Per andare a capo in modo forzato usare <newline>
   Questa frase inizia sempre al margine sinistro.

<sect>Elenchi
<p>Questo mette elementi dentro ad un elenco puntato con un segno grafico
   all'inizio di ogni elemento. Gli elenchi iniziano con il tag
   "itemize".

<itemize>
<item>Questo è il primo elemento di un elenco.
<item>Questo è il secondo elemento.
       <itemize>
       <item>Sono supportati livelli multipli (annidati).
       <item>Il secondo elemento in questo sottoelenco.
       </itemize>
       <enum>
       <item>Funzionano anche elenchi numerati usando <tt/enum/.
       <item>Questo è l'elemento numero 2.
       </enum>
<item>Questo è l'ultimo elemento nell'elenco principale.
</itemize>
</article>
```

5.5 Guida di consultazione rapida di LinuxDoc

5.5.1 Intestazione

```
<!doctype linuxdoc system>
<article>
<title>Guida di consultazione rapida
<author>David S. Lawyer
<date>v1.0, July 2000
<abstract>Qui va messo il sommario </abstract>
<toc> <!-- Commento: toc = Indice -->
```

5.5.2 Impaginazione del corpo

```
<sect>Capitolo 1            Nota: si metta un <p> sulla prima riga di
<sect1>Sottosezione 1.1     ogni sezione (o sottosezione, ecc.)
<sect1>Sottosezione 1.2
<sect>Capitolo 2            Si scelgano nomi di titoli da sostituire a
<sect1>Sottosezione 2.1     "Capitolo", "Sottosezione", ecc.
<sect2>Sotto-sottosezione 2.1.1
<sect2>Sotto-sottosezione 2.1.2
<sect1>Sottosezione 2.2
</article>
```

5.5.3 Tipi di carattere

Ci sono due modi per ottenerli:

```
<bf>carattere grassetto</bf> <em>carattere enfatizzato</em> <sf>carattere sans serif</sf>
<sl>carattere inclinato</sl> <tt>carattere a spaziatura fissa</tt> <it>carattere corsivo</it>
oppure:
<bf/grassetto/ <em/enfatizzato/ <sf/sans serif/
<sl/inclinato/ <tt/spaziatura fissa/ <it/corsivo/
```

5.5.4 Elenchi (è possibile l'annidamento)

```
Normali elenchi non numerati:              Elenchi numerati:
<itemize>                            <enum>
<item>Primo elemento                 <item>Primo elemento
<item>Secondo elemento               <item>Secondo elemento
<item>eccetera                       <item>eccetera
</itemize>                           </enum>
```

5.5.5 Collegamenti

```
Riferimenti incrociati:               Un collegamento email:
<ref id="links_" name="Collegamenti">   <url url="mailto:bob@tldp.org">
```

5.5.6 A capo, verbatim, URL

```
Per andare a capo forzatamente: <newline>
<tscreen><verb>
<url url="http://www.tldp.org">
<url url="http://www.tldp.org" name="Linux Documentation Project">.
</verb></tscreen>
```

5.5.7 Codici di carattere (macro)

Non è sempre necessario usarli.

- Usare `&` per la e commerciale (&).
- Usare `<` per una parentesi angolare sinistra (<).
- Usare `>` per una parentesi angolare destra (>).
- Usare `&etago;` per una parentesi angolare sinistra con una sbarra (</).

L'uso di questi è opzionale e io li uso raramente.

- Usare ‘‘ e ’’ per aprire e chiudere apici doppi.
- Usare `` per un soft-hyphen (trattino debole), per indicare cioè che questo è un buon punto per spezzare una parola molto lunga e inserire un trattino per la giustificazione orizzontale della riga.

Usare i seguenti solo se si hanno problemi in LinuxDoc con essi o se non si riesce ad ottenerli nel documento formattato. Io ho dovuto usarli raramente.

- Usare `$` per il simbolo del dollaro ($).
- Usare `#` per il simbolo cancelletto (#).
- Usare `%` per il simbolo di percentuale (%).
- Usare `˜` per la tilde (˜).
- Usare `&dquot;` per .

6 Ottenere/Usare il software LinuxDoc

Si potrebbe scrivere un documento LinuxDoc senza avere alcun software LinuxDoc. Tuttavia è assai probabile che esso contenga errori nei tag (o nel loro uso), così da vederselo restituire per le correzioni. Anche se non ci fossero errori, il risultato potrebbe non apparire nel modo esatto. Perciò è meglio avere sul proprio computer il software per convertire il proprio file sorgente.

La distribuzione di Linux Debian ha un pacchetto linuxdoc-tools. Esiste anche un pacchetto rpm per distribuzioni diverse da Debian. Precedentemente si chiamava sgml-tools; non si utilizzi il pacchetto sgmltools-2 perché è destinato principalmente per DocBook.

Per usare linuxdoc-tools, si eseguono i programmi di conversione sui file *.sgml. Per esempio, per ottenere un file di testo con una versione successiva alla 0.9.21-0.8 digitare: sgml2txt -f –blanks=1 mio-HOWTO.sgml. Per le versioni precedenti, a causa di un bug, si deve sostituire -f con –pass=-P-cbou. (Se interessati a maggiori informazioni su questo bug, guardare 11.1 (Il vecchio problema con le sequenze di escape nell'output in testo semplice).) Per ottenere output html digitare: sgml2html mio-HOWTO.sgml. Se appaiono degli errori, saranno mostrati il numero della riga e della colonna corrispondenti alla posizione dell'errore nel file sorgente. Digitando man -k sgml dovrebbe apparire un certo numero di altri programmi con una descrizione di una riga per ciascuno, ma non tutti sono per linuxdoc-sgml.

7 Errori e messaggi di errore

7.1 Errori che non creano messaggi di errore

Un errore importante è quello di dimenticare di mettere un tag <p> dopo l'intestazione di una sezione. In quel caso non c'è una fine dell'intestazione della sezione: tutto il paragrafo successivo diventa parte dell'intestazione e viene inserito nell'indice. LinuxDoc dovrebbe essere migliorato per poter individuare questo tipo di errori; per farlo manualmente, si guardi l'indice in formato testuale o html. Per correggere basta inserire il tag p mancante.

7.2 Messaggi di errore

Quando si esegue un programma linuxdoc (sgml2html o sgml2txt, per esempio) è possibile che si ottengano messaggi di errore. È necessario modificare il documento e correggere gli errori.

Un errore comune è la mancanza delle virgolette finali. Se si ottiene un messaggio di errore che non ha senso e si nota che nella posizione corrispondente all'errore sono presenti delle virgolette ( ) che sono corrette, allora è probabile che l'errore consista nell'aver in precedenza inserito una virgoletta di apertura a cui non ne corrisponde una di chiusura. Ad esempio <.... id=mia pagina web >, così che linuxdoc pensa che la virgoletta successiva, che può anche essere molti paragrafi dopo la posizione della virgoletta mancante, sia la virgoletta di chiusura. Dopo la falsa virgoletta di chiusura si aspetta quindi un carattere > che chiuda il tag, ma non lo trova. Per trovare e correggere la virgoletta mancante, si ricerchi una precedente.

Un unico errore può causare molti messaggi di errore. Nell'esempio precedente, vari tag possono essere contenuti all'interno delle virgolette sbagliate mentre non avrebbero dovuto essere all'interno di alcuna virgoletta. Di conseguenza linuxdoc non li troverà; ciò fa sì che non sappia che un certo tag è stato aperto e, se trova un tag di chiusura, dirà che un certo tag non è stato aperto. Per esempio, se mancasse il tag <itemize>, allora i tag <item> non avrebbero senso per linuxdoc ed esso riporterà un errore per ciascuno di questi tag trovato dopo la falsa virgoletta di chiusura.

È bene sapere che l'uso delle maiuscole o minuscole per i nomi dei tag non è importante, perciò i messaggi di errore saranno mostrati con i nomi dei tag in maiuscolo anche se sono stati inseriti in lettere minuscole.

Capire al meglio i messaggi di errore richiede la conoscenza del gergo relativo all'sgml, il che non è veramente necessario a meno che non si ottengano errori che non si riescono a correggere e dei quali non si capisce il messaggio di errore. La sezione seguente tratta del gergo sgml.

8 Gergo nei messaggi di errore

8.1 Introduzione

Non è veramente necessario leggere questa sezione a meno che non si abbiano problemi oppure non si abbia la curiosità di sapere come funzionano l'sgml e linuxdoc. I messaggi di errore possono contenere parole come element (elemento), entity (entità, o entities al plurale), attribute (attributo), literal (letterale) e delimiter (delimitatore). Vari elementi, entità e attributi sono definiti per linuxdoc nel Data Type Definition (o DTD) per LinuxDoc. Il DTD non li definisce a parole, ma usa un formato piuttosto criptico per definire la loro sintassi (ma non la loro semantica).

8.2 Gli elementi

Un elemento è un qualcosa come un tag, ma è un concetto molto più ampio. Gli elementi esistono non solo in LinuxDoc, ma in tutti i linguaggi sgml, come ad esempio html. L'intero documento LinuxDoc è diviso in elementi, ma questi sono annidati; questo significa che alcuni elementi possono comparire all'interno di altri elementi. Se si usa il tag <article> per il proprio documento, allora tutto il documento è l'elemento <article>, tranne per il primissimo tag che dice che ciò che segue è LinuxDoc. All'interno di tale elemento article sono annidati molti altri elementi.

Ogni paragrafo, per esempio, è un elemento, anche se i paragrafi sono separati tra di loro da una riga vuota invece che da tag. C'è però un tag implicito che racchiude ogni paragrafo ed il software che analizzerà il documento LinuxDoc di fatto inserirà questi tag mancanti. Inserirà anche i tag finali (di chiusura) dove non era obbligatorio scriverli. In questo modo LinuxDoc fa risparmiare molto tempo. Un elemento consiste perciò in un tag di apertura, in uno di chiusura (corrispondente a quello di apertura) e a tutto ciò che è racchiuso tra di essi (compresi spesso altri elementi con i loro tag). Si noti che i tag anche se omessi sono tuttavia sempre implicitamente presenti. In alcuni casi un tag non racchiude alcunché, come un tag URL per un collegamento Internet; questi tag sono essi stessi elementi. All'interno dell'elemento article si trovano elementi sect (sezioni) che iniziano con <sect>. All'interno degli elementi sect ci sono poi spesso elementi sect1, ecc.

Ci sono alcuni casi in cui è presente un elemento, ma è opzionale l'uso sia del tag di apertura, sia di quello di chiusura. Perciò anche se tale tag non è presente nel documento, le parti del documento che avrebbbe dovuto racchiudere sono comunque elementi di quei tag mancanti.

Un'entità è come una definizione di macro. Per esempio, si potrebbe definire il nome elenco come significativo dei vari tipi di elenchi. Questa parola elenco viene poi usata nel DTD solo per specificare, per esempio, che è possibile avere un elenco all'interno di un paragrafo. È semplicemente un'abbreviazione per l'autore di un DTD. Questo tipo di entità non è mai usato in un documento LinuxDoc. Esiste però anche un altro tipo di entità che può essere usata all'interno di un documento ed è quella che definisce un carattere speciale come &etago per </ (inizio del tag di chiusura). Lo si usa, per esempio, quando si desidere mettere </article> all'interno di una frase per spiegare cosa significa, in modo che il software che converte il LinuxDoc non pensi che sia davvero la fine dell'articolo.

8.3 Literal e delimitatori

Un literal è il nome di qualcosa, come il nome su cui si clicca in un collegamento html; può essere formato da una o più parole. Un delimitatore è ciò che separa qualcosa da qualcos'altro. Per una citazione l'ultima è il delimitatore di chiusura. Perciò per name=mio sito web, il literal è 'mio sito web' e i delimitatori di questo literal sono i due caratteri : il primo è il delimitatore di apertura e il secondo il delimitatore di chiusura. Così la frase manca il delimitatore di chiusura di un literal significa che ci si è dimenticati di inserire un finale dopo un nome.

9 Scrivere l'HOWTO

9.1 Prima di iniziare a scrivere

Per prima cosa ci si iscriva alla lista di discussione andando all'indirizzo `www.en.tldp.org` e si sottoponga la propria proposta a questa lista. Se si prende in carico un HOWTO non più mantenuto, si contatti l'autore precedente. Ciò può essere richiesto dal copyright/licenza, ma andrebbe fatto in ogni caso come forma di cortesia.

9.2 Linee guida

Queste sono principalmente opera di Tim Bynum (un ex-coordinatore degli HOWTO).

- Assicurarsi di usare un formato accettato (come il LinuxDoc :-).
- Cercare di usare una struttura e un'organizzazione significativa, e scrivere chiaramente. Ricordare che molte delle persone che leggono gli HOWTO non parlano l'inglese come loro prima lingua.
- Assicurarsi che tutte le informazioni siano corrette. Non insisterò mai abbastanza su questo punto. Nel dubbio, si facciano delle congetture, ma si sia chiari sul fatto che si stanno facendo solamente delle ipotesi. Io uso ?? se non sono sicuro.
- Assicurarsi di trattare la più recente versione del software disponibile.
- Prendere in considerazione l'inclusione di una sezione FAQ oppure delle sezioni Problemi comuni o Risoluzione di problemi.
- Assicurarsi di mettere nel copyright il proprio nome e di includere una licenza che soddisfi i requisiti dichiarati nel manifesto di LDP.
- Usare l'intestazione standard con il titolo, l'autore, la data (includendo un numero di versione). Si veda 5.4 (Esempio 3).
- Da ultimo, prepararsi a ricevere messaggi di posta elettronica con domande e commenti dei lettori. Sta alla propria discrezione quanto aiutare gli altri, ma si dovrebbero ascoltare i buoni suggerimenti e le segnalazioni di errori. Si potrebbero anche ricevere dei messaggi di ringraziamento.

9.3 Sottoporre l'HOWTO, ecc.

Dopo aver scritto l'HOWTO, inviare via posta elettronica il sorgente SGML all'indirizzo: submit@tldp.org. Successivamente tutto ciò che è necessario fare è tenere aggiornato l'HOWTO sottoponendo periodicamente gli aggiornamenti allo stesso indirizzo di posta usato per la prima edizione.

10 Ulteriori informazioni

C'è un HOWTO, il Linuxdoc Reference, che tratta LinuxDoc in modo molto più dettagliato di quanto non faccia questo mini-HOWTO.

11 Appendice

11.1 Il vecchio problema con le sequenze di escape nell'output in testo semplice

Prima della versione 0.9.21-0.8 c'era un bug nell'output in formato di testo semplice. Per sgml2txt, era necessaria l'opzione –pass=-P-cbou per ottenere l'output in puro testo dato che altrimenti, se si usava l'opzione -f, si otteneva un output di testo che metteva l'enfasi sulle parole e sulle lettere usando sequenze di escape e sovrascrittura. L'esempio di un pallino creato con la sovrascrittura è +ˆHo che in una stampante stamperebbe +, poi tornerebbe indietro di uno spazio (ˆH, backspace) e quindi stamperebbe o sopra al + esistente. Questo non sembra funzionare sui terminali (non possono sovrascrivere). Notare che anche se si ha la versione più recente, si ottiene sempre questo output non voluto se non si usa l'opzione -f.

In caso si sia interessati ai dettagli, –pass passa l'opzione -P-cbou al programma groff (usato da sgml2txt) e l'opzione -P di groff passa le opzioni -cbou a grotty (un post-elaboratore per groff), forzandolo a generare solamente output in puro testo. Vedere la pagina di manuale di grotty. In breve, -c evita le sequenze di escape, ma permette la sovrascrittura, -bou tuttavia proibisce la sovrascrittura quando è usata l'opzione -c. Il risultato è l'assenza di sovrascritture e di sequenze di escape nell'output. -b proibisce la sovrascrittura per far apparire un carattere in grassetto, -u previene l'uso della sovrascrittura per sottolineare e -o proibisce altri tipi di sovrascrittura quale l'esempio del pallino sopra citato. Un modo alternativo per eliminare la sovrascrittura è di usare l'opzione -f con sgml2txt, ma è sempre necessario passare l'opzione -c a grotty per eliminare le sequenze di escape, a meno che non si abbia la versione più recente.

Che confusione! L'opzione predefinita dovrebbe probabilmente essere il puro testo, così da rendere tutte queste opzioni non necessarie. Finalmente sono riuscito a far correggere tutto ciò, perciò da metà 2007 in poi, si può usare solamente l'opzione -f invece di –pass=-P-cbou. Se si ottengono queste sequenze di escape e sovrascritture nel proprio file di output, ma si usa il comando Linux cat per visualizzare il testo, il risultato è molto bello. L'uso di paginatori o di editor sul file output di testo tuttavia ha spesso come risultato danni ai caratteri di escape, così che si vedono nel proprio testo un mucchio di caratteri non desiderati, che avrebbero dovuto far parte delle sequenze di escape. In alcuni casi i paginatori possono mostrare alcune sovrascritture correttamente, ma gli editor (come vim) non lo fanno. Eliminare tutte le sovrascritture permette perciò di usare un qualsiasi editor o paginatore per leggere.