## 53.1 Sistemi SGML basati su Qwertz

Il sistema standard utilizzato inizialmente per la documentazione dei sistemi GNU/Linux si è basato sul DTD Qwertz, dal quale hanno avuto origine una serie di derivazioni e di strumenti di composizione SGML che hanno avuto una certa importanza. Nelle sezioni successive si mostra solo il funzionamento essenziale di alcuni di questi strumenti di composizione; in particolare LinuxDoc e DebianDoc.

Tra tutte le derivazioni di strumenti di composizione SGML basati originariamente sul DTD Qwertz, DebianDoc sembra essere stato il sistema più coerente, con il quale si garantiva la composizione tipografica dal formato PostScript alla pagina di manuale pura e semplice.

### 53.1.1 Struttura di LinuxDoc

La struttura di un sorgente SGML secondo il DTD LinuxDoc è generalmente la seguente:

```
<!DOCTYPE linuxdoc SYSTEM>
<article>
<titlepag>
<title>Titolo del documento</title>
<author>
        <name>Pinco Pallino ppallino@dinkel.brot.dg</name>
</author>
<date>29/02/1999</date>
<abstract>
Breve introduzione al documento.
</abstract>
</titlepag>
<toc>
<sect>Prima sezione
<p>
Contenuto della prima sezione,
...
...
(eventuali altre sezioni)
</article>
```

Con l'istruzione '**<!DOCTYPE linuxdoc SYSTEM>**' si afferma di voler utilizzare il DTD '**linuxdoc**'. Il documento è delimitato dall'elemento '**article**' che rappresenta uno tra i diversi tipi di struttura possibile del documento. Il DTD LinuxDoc è derivato dal Qwertz che è strutturato in modo da imitare il comportamento di LaTeX. In questo modo, nel DTD originale sono previste diverse strutture, tutte riferite ad analoghi tipi di documento LaTeX. La tendenza generale è quella di utilizzare sempre solo la struttura '**article**', soprattutto perché lo scopo di SGMLtools è stato quello di permettere la trasformazione del sorgente SGML in un grande numero di altri formati, non solo LaTeX.

Dopo l'inserimento dell'elemento '**title**' e di tutto ciò che deve contenere (titolo, autore, descrizione del documento), è possibile inserire il marcatore '**<toc>**', con il quale si intende ottenere un indice generale.

Dopo l'indice generale inizia il testo del documento, suddiviso in sezioni, il cui inizio è evidenziato dai marcatori: '**<sect>**', '**<sect1>**', '**<sect2>**'.

#### 53.1.1.1 Utilizzo sommario

Attraverso SGMLtools, si ottiene un documento finale a partire da un sorgente SGML. Per questo, si elabora il sorgente come si fa con un linguaggio di programmazione durante la compilazione. La prima fase è il controllo di validità.

```
sgmlcheck sorgente_sgml
```

Una volta verificata la correttezza formale dal punto di vista del DTD, si può richiedere la trasformazione in un altro formato. Nell'elenco seguente vengono mostrati solo alcuni tipi di trasformazione, i più importanti. In effetti non tutto funziona nello stesso modo e alcuni tipi di conversioni sono difettosi.

Quando si progetta di realizzare un documento attraverso SGMLtools/LinuxDoc, è importante decidere subito quali formati devono essere ottenuti necessariamente, in modo da poter controllare il loro funzionamento dall'inizio dell'opera. Per esempio, il fatto che si riesca a ottenere un formato PostScript corretto, non garantisce che gli altri formati generino un risultato altrettanto buono.[1]

La conversione in LaTeX si ottiene facilmente attraverso il comando seguente:

```
sgml2latex --output=tex sorgente_sgml
```

Viene generato un file con lo stesso nome del sorgente, terminante con l'estensione '`.tex`'. Questo file contiene riferimenti a stili addizionali che fanno parte del pacchetto SGMLtools. Questo fatto deve essere tenuto in considerazione se si vuole poi rielaborare questo file con LaTeX.

La composizione del documento in PostScript avviene attraverso l'elaborazione successiva da parte di LaTeX, richiamato automaticamente da SGMLtools.

```
sgml2latex --output=ps sorgente_sgml
```

Quello che si ottiene è un file con lo stesso nome del sorgente, terminante con l'estensione '`.ps`'.

La conversione in formato HTML viene gestita completamente all'interno di SGMLtools, attraverso il sistema di programmi in Perl che lo compongono.

```
sgml2html sorgente_sgml
```

Si ottengono dei file HTML collegati attraverso riferimenti ipertestuali.



#### 53.1.1.2 Supporto per altri SGML

SGMLtools ha un supporto limitato per HTML. Precisamente, consente di verificare un file HTML attraverso il DTD HTML 3.2. Si può usare il comando seguente, che è lo stesso visto nel caso dei file SGML.

```
sgmlcheck sorgente_html
```

'**sgmlcheck**' determina da solo che si tratta di un file HTML. Comunque, un file HTML corretto dovrebbe iniziare con la dichiarazione seguente:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
```

Eventualmente, sono ammissibili anche altre forme,

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
```

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Draft//EN">
```

dove '**Draft**' si riferisce in particolare alla prima stesura della versione 3.2.

Si può osservare che un file HTML apparentemente corretto dato il risultato che si ottiene con il programma usato per visualizzarlo, può contenere un gran numero di errori formali secondo il suo DTD.

### 53.1.2 LinuxDoc più in dettaglio

Lo standard LinuxDoc, come suggerisce il nome, è quello che si è utilizzato originariamente per la documentazione di GNU/Linux. Del DTD relativo, '`linuxdoc.dtd`', vengono sfruttate ufficialmente solo alcune delle caratteristiche. Per esempio, la definizione dell'incorporazione di immagini e le tabelle sono rimaste come eredità dallo standard Qwertz, ma il loro utilizzo va evitato, preferendo piuttosto l'uso di strumenti SGML basati su DocBook.

#### 53.1.2.1 Preambolo e definizione dello stile

Come accennato all'inizio del capitolo, un documento LinuxDoc inizia con un preambolo che descrive il tipo di documento ('**linuxdoc**' appunto), lo stile (in questo caso '**article**'), il titolo, l'autore e altre informazioni eventuali.

```
<!DOCTYPE linuxdoc SYSTEM>

<article>

<titlepag>
<title>Il mio primo articolo</title>
<author>Pinco Pallino, pincop@dinkel.brot.dg</author>
<date>v0.01, 29 febbraio 1999</date>
<abstract>
Breve anticipazione del contenuto del documento.
</abstract>
</titlepag>

<toc>

<sect>Prima sezione
<p>
Contenuto della prima sezione.

</article>
```

Dopo il preambolo può essere collocato un indice generale che viene costruito automaticamente attraverso l'elemento '**toc**'. Quindi si può iniziare il corpo del documento suddiviso in sezioni. Al termine, la chiusura dello stile dichiarato nel preambolo definisce la fine del documento.

Lo stile '**article**' è quello standard per i documenti LinuxDoc ed è anche quello raccomandato. Consente la suddivisione del documento per sezioni e non per capitoli. Viene chiuso alla fine del documento.

### 53.1.2.2 Suddivisione del documento

A seconda dello stile di documento utilizzato, la suddivisione del contenuto può avvenire in modi differenti. In pratica, utilizzando lo stile '**article**', la suddivisione avviene solo per sezioni, identificate dall'elemento '**sect**'.

1. '**sect**'
2. '**sect1**'
3. '**sect2**'

Ciò significa che una sezione '**sect**' può scomporsi in sottosezioni '**sect1**', che a loro volta si possono scomporre in altre sottosezioni di livello inferiore '**sect2**', ecc. In generale, se possibile, è conveniente limitarsi soltanto a due livelli di suddivisione.

```
<!DOCTYPE linuxdoc SYSTEM>

<article>

<titlepag>
<title>Il mio primo articolo</title>
<author>Pinco Pallino, pincop@dinkel.brot.dg</author>
<date>v0.01, 29 febbraio 1999</date>
<abstract>
Breve anticipazione del contenuto del documento.
</abstract>
</titlepag>

<toc>

<sect>Prima sezione
<p>
Contenuto della prima sezione.
...

<sect1>Una sottosezione
<p>
Contenuto della sottosezione.
...

<sect>Seconda sezione
<p>
...
...

</article>
```

L'ambiente delimitato da una sezione di qualunque livello, non richiede l'indicazione esplicita della sua conclusione. È invece necessaria l'inserzione dell'indicazione dell'inizio di un paragrafo, subito dopo il titolo della sezione stessa. L'esempio mostrato sopra dovrebbe chiarirne il funzionamento.

### 53.1.2.3 Paragrafi

Il testo di un documento normale è suddiviso in paragrafi. L'indicazione dell'inizio o della conclusione di un paragrafo è facoltativa. È sufficiente staccare i paragrafi con almeno una riga bianca per dare questa informazione a LinuxDoc. Resta comunque possibile l'indicazione esplicita dei paragrafi attraverso l'elemento '**p**'. È obbligatoria l'indicazione dell'inizio del primo paragrafo di una sezione, perché non esiste altro modo per capire quando finisce il titolo (della sezione) e quando inizia il testo.

### 53.1.2.4 Elenchi

Si hanno a disposizione tre tipi di elenchi: descrittivo ('**descrip**'), puntato ('**itemize**') e numerato ('**enum**').

L'elenco descrittivo è definito dall'elemento '**descrip**'. Le parti descrittive di questo elenco sono costituite da elementi '**tag**'. Ciò che è contenuto all'interno della sequenza '**<tag>**...**</tag>**' appare evidenziato in un'unica riga e generalmente non può contenere simboli particolari (dipende dal tipo di trasformazione che si vuole ottenere). Per esempio:



```
<descrip>
<tag>primo</tag>primo elemento;
<tag>secondo</tag>secondo elemento;
<tag>terzo</tag>terzo elemento.
</descrip>
```

genera l'elenco seguente:

```
primo
    primo elemento;
secondo
    secondo elemento;
terzo
    terzo elemento.
```

L'elenco puntato è costituito dall'elemento '**itemize**' che si articola in elementi '**item**', che in pratica costituiscono le varie voci dell'elenco. Per esempio:

```
<itemize>
<item>primo elemento;
<item>secondo elemento;
<item>terzo elemento.
</itemize>
```

genera l'elenco puntato seguente:

```
* primo elemento;
* secondo elemento;
* terzo elemento.
```

L'elenco numerato è costituito dall'elemento '**enum**' che si articola in elementi '**item**', come nel caso dell'elenco puntato. Per esempio:

```
<enum>
<item>primo elemento;
<item>secondo elemento;
<item>terzo elemento.
</enum>
```

genera l'elenco numerato seguente:

```
1 primo elemento;
2 secondo elemento;
3 terzo elemento.
```

Generalmente, se il tipo di composizione finale lo consente, gli elenchi possono essere annidati e contenere anche testo normale che viene rappresentato allineato in modo opportuno.

```
<descrip>
<tag>primo</tag>
        Primo elemento descrittivo.

        Continuazione del primo elemento descrittivo.

<tag>secondo</tag>
        Secondo elemento descrittivo.

        <enum>
        <item>Prima suddivisione.

                <enum>
                <item>Ulteriore suddivisione.
                <item>Ancora un altro punto.
                </enum>

        <item>Seconda suddivisione.

                <itemize>
                <item>Ecco un sottoelenco puntato.
                <item>Un secondo elemento dell'elenco
                      puntato.
                </itemize>

        <item>Terza suddivisione.
        </enum>

<tag>terzo</tag>
        Terzo elemento descrittivo.

</descrip>
```

L'esempio sopra riportato si traduce in qualcosa che è simile a ciò

che segue:

```
primo
    Primo elemento descrittivo.
    Continuazione del primo elemento descrittivo.
secondo
    Secondo elemento descrittivo.
    1  Prima suddivisione.
        a  Ulteriore suddivisione.
        b  Ancora un altro punto.
    2  Seconda suddivisione.
        *  Ecco un sottoelenco puntato.
        *  Un secondo elemento dell'elenco puntato.
    3  Terza suddivisione.
terzo
    Terzo elemento descrittivo.
```

#### 53.1.2.5 Inclusione di testo letterale

Si incontra spesso la necessità di includere in un documento del testo letterale. In generale si tratta di listati di programma o cose simili che possono contenere caratteri o simboli che di solito dovrebbero essere scritti utilizzando dei codici macro particolari. Per questo si utilizza l'elemento '**verb**'.

Al suo interno è consentito includere un testo che deve essere riprodotto esattamente com'è, spazi e caratteri strani inclusi, utilizzando, quando possibile, lo stesso carattere usato per il testo normale. Per quanto riguarda la libertà di inclusione di simboli, esiste comunque una piccola limitazione:

- il simbolo '**&**' può essere inserito solo con un codice macro '**&ero;**' (mentre nel testo normale si usa la macro '**&**');
- la sequenza di simboli minore+barra obliqua ('**</**'), usata di solito per iniziare l'indicazione di un marcatore conclusivo, deve essere rappresentata usando il codice macro '**&etago;**'.

Di solito, il testo contenuto in questo elemento è preferibile che appaia in un carattere dattilografico. Per questo, generalmente, '**verb**' viene a sua volta inserito in un elemento '**tscreen**'.

```
<tscreen><verb>
Ecco un testo che contiene strani simboli # \ [ ].
</verb></tscreen>
```

#### 53.1.2.6 Testo citato

Quando si cita del testo o si vuole fare risaltare una nota, si usano rientri e tipi di carattere diversi. Gli elementi utilizzati per questo scopo sono: '**quote**' e '**tscreen**'.

All'interno dell'elemento '**tscreen**' il testo viene riportato tutto con caratteri a larghezza fissa e rientrato leggermente. Di solito viene usato per incorporare l'elemento '**verb**', in modo da poter inserire simboli particolari senza la necessità di doverli convertire.

```
<tscreen>
Ecco del testo riportato con carattere a larghezza fissa
o dattilografico.
</tscreen>
```

L'elemento '**quote**' fa in modo di rientrare leggermente il testo, per fare risaltare che si tratta di una citazione.

```
<quote>
Senza nessuna precisazione, i documenti Linux HOWTO hanno
il copyright dei loro rispettivi autori. I documenti Linux
HOWTO possono essere riprodotti e distribuiti, completi o
in...
</quote>
```

#### 53.1.2.7 Enfatizzazioni

All'interno di un testo normale è possibile intervenire per modificare l'aspetto del carattere. Generalmente, qualsiasi intervento verso la definizione dell'aspetto del risultato finale è inopportuno in un sorgente SGML. Infatti, SGML dovrebbe servire per definire gli oggetti che compongono il testo e il documento in generale; quindi, è compito dei programmi di conversione attribuire un aspetto particolare al risultato finale.



LinuxDoc consente ancora di intervenire sull'aspetto di alcune parti di testo, attraverso l'indicazione di testi in corsivo, neretto e dattilografico. Resta tuttavia da considerare che queste possibilità sono destinate a scomparire, in favore di una definizione più precisa delle componenti del testo.

L'elemento '**bf**' si utilizza per rendere in neretto il testo racchiuso.

```
Esempio di un testo in <bf>neretto o bold face</bf>.
```

L'elemento '**it**' si utilizza per rendere in corsivo il testo racchiuso.

```
Esempio di un testo <it>corsivo</it>.
```

L'elemento '**tt**' si utilizza per rendere in carattere dattilografico il testo racchiuso.

```
Esempio di un testo <tt>a larghezza fissa o
dattilografico</tt>.
```

#### 53.1.2.8 Riferimenti incrociati

Possono essere fatti dei riferimenti interni o esterni al documento. Generalmente, all'interno del documento si utilizza l'elemento '**label**' come segnaposto e l'elemento '**ref**' come puntatore. Per fare dei riferimenti all'esterno del documento, si fa uso dell'elemento '**url**' oppure di '**htmlurl**'.

Un'etichetta, definita attraverso l'elemento '**label**', permette di marcare una posizione nel documento a cui si vuole poter fare riferimento. Si tratta di un elemento vuoto che contiene un attributo obbligatorio: '**ID**'. Questo attributo contiene il valore dell'etichetta che identifica quindi la posizione che si vuole marcare.

```
<sect>Note personali<label id="note1">
<p>
        bla bla bla bla...
```

L'esempio mostra un possibile uso di '**label**' per marcare l'inizio di una sezione. In linea di massima, un'etichetta di questo genere permette di fare riferimenti di due tipi: la pagina in cui si trova e il numero della sezione o dell'oggetto, in relazione al contesto in cui si trova. Un'etichetta può apparire nei contesti seguenti:

- all'interno di testo normale, facendo riferimento al capitolo e alla sezione in cui si trova;
- all'interno di un elemento '**caption**' di una figura, facendo riferimento al numero della figura;
- all'interno di un elemento '**caption**' di una tabella, facendo riferimento al numero della tabella.

È importante che queste etichette-segnaposto non contengano caratteri strani, altrimenti il programma di composizione potrebbe non gestirle correttamente.

Un elemento '**ref**' si comporta come puntatore o riferimento a un'etichetta definita attraverso l'elemento '**label**'. All'interno di un documento stampato genera un riferimento numerico che dipende dal contesto in cui si trova l'etichetta (il numero della sezione, della figura o della tabella), mentre in un documento HTML genera un riferimento ipertestuale (*link*).

Si tratta di un elemento vuoto che contiene un attributo obbligatorio, '**ID**', e uno opzionale, '**NAME**'. L'attributo '**ID**' contiene il nome dell'etichetta a cui si intende fare riferimento, l'attributo '**NAME**' viene inserito per dare un nome al riferimento che viene creato quando si genera un documento HTML.

```
Vedere la sezione <ref id="linuxdoc-xref-ref"
name="riferimento">.
```

Un elemento '**pageref**' di comporta come puntatore o riferimento a un'etichetta. All'interno di un documento stampato genera un riferimento al numero della pagina che contiene l'etichetta.[2]

Si tratta di un elemento vuoto che contiene un attributo obbligatorio, '**ID**', destinato a contenere il nome dell'etichetta a cui si intende fare riferimento.

Un elemento '**url**' si comporta come riferimento a un URI. All'interno di un documento stampato genera la rappresentazione di

questo indirizzo URI, mentre in un documento HTML crea un riferimento ipertestuale vero e proprio. Un elemento '**htmlurl**' si comporta in maniera analoga, ma non riporta l'indirizzo URI nel documento stampato.[3]

Si tratta di elementi vuoti che contengono un attributo obbligatorio, '**URL**', destinato a indicare l'indirizzo URI a cui si intende fare riferimento, e uno opzionale, '**NAME**'. Si osservi la differenza tra i due tipi di puntatori attraverso l'esempio seguente:

```
<url url="http://ildp.psy.unipd.it/" name="ILDP">
&egrave; il progetto di documentazione di Linux in italiano.

<htmlurl url="http://ildp.psy.unipd.it/" name="ILDP">
&egrave; il progetto di documentazione di Linux in italiano.
```

Nel primo caso, assieme al valore dell'attributo '**NAME**' viene visualizzato anche l'URI, mentre nel secondo viene mostrato solo il valore di '**NAME**'.

L'elemento '**footnote**' permette di inserire una nota che da stampare a piè di pagina (ma potrebbe non funzionare correttamente nella composizione in HTML).

```
LinuxDoc &egrave; una derivazione di
Qwertz<footnote>Il nome della tastiera tedesca.</footnote>.
```

### 53.1.2.9 Indici

Il sistema è in grado di generare automaticamente l'indice generale del documento e un indice analitico. Per ottenere l'indice generale è sufficiente inserire l'elemento '**toc**' (vuoto) subito dopo il preambolo. L'esempio seguente mostra in che modo si può inserire un indice di questo tipo.

```
<!DOCTYPE linuxdoc SYSTEM>

<article>

<titlepag>
<title>Il mio primo articolo</title>
<author>Pinco Pallino, pincop@dinkel.brot.dg</author>
<date>v0.01, 29 febbraio 1999</date>
<abstract>
Breve anticipazione del contenuto del documento.
</abstract>
</titlepag>

<toc>

<sect>Prima sezione
<p>
Contenuto della prima sezione.

</article>
```

Ogni tipo di conversione in un formato finale del documento SGML gestisce la generazione dell'indice generale a modo proprio. Di solito sono garantiti solo due livelli di titoli (sezioni).

L'indice analitico potrebbe essere disponibile solo per la conversione attraverso LaTeX. Si ottiene marcando alcune porzioni di testo attraverso l'elemento '**nidx**', oppure '**ncdx**', come nell'esempio seguente:

```
<sect>Pallini e sfere<nidx>pallino</nidx><ncdx>sfera</ncdx>
<p>
Questa sezione tratta di pallini e sfere in generale, fino a
giungere alla descrizione dei cuscinetti a
sfera.<nidx>cuscinetto a sfera</nidx>
```

Quanto contenuto all'interno degli elementi '**nidx**' e '**ncdx**' non viene a fare parte del testo; tutte le conversioni che non possono farne uso lo trattano come un commento da ignorare. La conversione in LaTeX genera corrispondentemente il comando LaTeX '**\index{…}**', ma nel caso particolare di '**ncdx**', vengono aggiunti dei codici di composizione in modo tale che nell'indice la stringa corrispondente appaia evidenziata con un testo dattilografico.

Per usare in pratica l'indice analitico, occorrono diverse fasi:



- la generazione del documento finale attraverso LaTeX;
- la generazione di un file indice, sempre attraverso LaTeX;
- la rielaborazione del file indice;
- la costruzione di un documento finale attraverso l'indice, in modo da poterlo abbinare al documento principale.

La generazione del file indice avviene attraverso il comando seguente:

```
sgml2latex --makeindex sorgente_sgml
```

Si ottiene un file, il cui nome ha la stessa radice del sorgente SGML e l'aggiunta dell'estensione '`.idx`'. Questo file deve essere rielaborato da '**makeindex**' che è un programma abbinato alle distribuzioni comuni di LaTeX.

```
makeindex < indice_generato > indice_rielaborato
```

Il file dell'indice rielaborato potrebbe avere la fisionomia dell'esempio seguente:

```
\begin{theindex}

  \item cuscinetto a sfera, 1
  \item cuscino, 15

  \indexspace

  \item pallino, 87
  \item pallone, 82
  \item pallottola, 54, 55
  \item pallottoliere, 50

  \indexspace

  \item {\tt sfera}, 30, 43
  \item steroide, 23

\end{theindex}
```

Per giungere a un risultato finale, cartaceo, occorre aggiungergli qualcosa in modo che diventi un documento LaTeX vero e proprio. Come nell'esempio seguente:

```
\documentclass[a4paper]{article}

\usepackage[italian]{babel}
\usepackage[utf8x]{inputenc}
\usepackage[T1]{fontenc}

\begin{document}

\begin{theindex}

  \item cuscinetto a sfera, 1
  \item cuscino, 15

  \indexspace

  \item pallino, 87
  \item pallone, 82
  \item pallottola, 54, 55
  \item pallottoliere, 50

  \indexspace

  \item {\tt sfera}, 30, 43
  \item steroide, 23

\end{theindex}

\end{document}
```

In tal modo, attraverso LaTeX si può passare alla trasformazione in un documento finale DVI; successivamente, attraverso '**dvips**', si può ottenere una trasformazione in PostScript.

```
latex documento_latex
```

```
dvips -o documento_ps documento_dvi
```

### 53.1.2.10 Inclusione di immagini

All'interno di un documento è possibile fare riferimento a immagini in formato EPS (*Encapsulated PostScript*), che vengono utilizzate nella trasformazione in PostScript attraverso LaTeX e '**dvips**'. Parallelamente è possibile fare anche riferimento a immagini (di solito equivalenti) in formati diversi, adatti alla trasformazione in HTML.

L'elemento '**figure**' racchiude le informazioni necessarie per l'inserzione di un'immagine. All'interno del marcatore di apertura è possibile specificare la posizione prescelta dell'immagine, per la trasformazione attraverso LaTeX, utilizzando l'attributo '**LOC**' (*location*). In pratica conviene quasi sempre utilizzare la stringa '**htbp**' che dice a LaTeX di collocare l'immagine nel posto più adatto, cominciando dalla posizione di partenza (*here*), quindi nella parte superiore della pagina (*top*), poi ancora nella parte inferiore (*bottom*) e infine, se ogni tentativo fallisce, in una pagina dedicata (*page*). Il valore predefinito di questo attributo è '**tbp**' con il significato che si può intuire.

```
<figure LOC=htbp>
        <eps file="esempio" height="5cm">
</figure>
```

L'esempio indica di visualizzare l'immagine '`esempio.ps`' collocata nella directory '`figure/`' a partire dalla posizione corrente.

L'elemento '**eps**' serve all'interno di un elemento '**figure**' per definire il file da visualizzare utilizzando l'attributo '**FILE**'. Questo file viene poi utilizzato nella composizione in PostScript attraverso LaTeX. Il nome del file che viene fornito **non** deve contenere l'estensione '`.ps`' che è sottintesa e obbligatoria. Un altro attributo obbligatorio è '**HEIGHT**', con cui si definisce l'altezza dell'immagine. L'esempio già mostrato in precedenza, specificava a questo proposito un'altezza di 5 cm. La larghezza viene regolata in proporzione.

L'elemento '**img**' serve invece a definire il file da visualizzare per la composizione in HTML. Anche in questo caso si utilizza l'attributo '**FILE**'. Al contrario del caso di '**eps**', il nome del file che viene fornito deve essere indicato completo di estensione.

```
<figure LOC=tbp>
        <eps file="esempio" height="5cm">
        <img src="figure/esempio.jpg">
</figure>
```

L'esempio indica di includere l'immagine '`esempio.ps`', per la composizione attraverso LaTeX, e '`esempio.jpg`' per quella in HTML.

L'elemento '**caption**' può essere usato all'interno della definizione di una figura per indicare la descrizione o il titolo della figura stessa. All'interno di questa descrizione si può inserire anche un'etichetta, l'elemento '**label**', in modo da permettere un riferimento al numero della figura all'interno del testo.

```
<figure LOC=tbp>
        <eps file="esempio" height="5cm">
        <img src="figure/esempio.jpg">
        <caption>
                <label id="figura-esempio">
                Immagine di esempio
        </caption>
</figure>
```

L'esempio inserisce la figura rappresentata dal file '`esempio.ps`', nel caso di trasformazione in LaTeX, oppure '`esempio.jpg`' in caso di trasformazione in HTML; inoltre appare una descrizione e un'etichetta per potervi fare riferimento.



### 53.1.2.11 Tabelle

All'interno di un documento è possibile inserire delle tabelle, ma questo solo se si intende trasformare il proprio documento in LaTeX. In HTML si riesce a ottenere qualcosa, ma decisamente scadente. Per questo motivo, l'uso delle tabelle deve essere riservato ai casi di effettiva necessità.

Le tabelle sono composte essenzialmente da righe separate da un separatore di riga, dove ogni riga è suddivisa a sua volta in colonne attraverso un separatore di colonna.

L'elemento '**table**' delimita la zona di descrizione di una tabella. All'interno del marcatore di apertura è possibile specificare la posizione prescelta della tabella, utilizzando l'attributo '**LOC**' (*location*), che si comporta nello stesso modo di quello utilizzato nell'elemento '**figure**'.

L'elemento '**tabular**', interno a '**table**', definisce le caratteristiche di una tabella. All'interno del marcatore di apertura è necessario specificare l'allineamento orizzontale del contenuto delle celle e la separazione di queste attraverso linee verticali. l'attributo utilizzato per questo è '**CA**' (*Column alignment*) e il suo valore consigliabile è una stringa composta da una serie di lettere '**l**', una per ogni colonna esistente nella tabella.

Le righe della tabella sono concluse dall'elemento '**rowsep**', mentre le colonne sono staccate l'una dall'altra attraverso l'elemento '**colsep**'. È possibile inserire una linea orizzontale di separazione utilizzando l'elemento '**hline**'. Tutti questi elementi di descrizione delle righe, sono vuoti.

Si osservi questo esempio. Si suppone di voler rappresentare una tabella di quattro righe, più una di intestazione, divisa in due sole colonne, secondo lo schema seguente:

```
----------------------------------------
Parametro LOC   Posizione corrispondente
----------------------------------------
h               posizione attuale
t               superiore
b               inferiore
p               pagina
----------------------------------------
        Esempio di tabella.
```

Il codice necessario è quello mostrato di seguito.

```
<table LOC=tbp>
<tabular colonne="2">
    <hline>
    Parametro loc <sepcol> Posizione corrispondente <rowsep>
    <hline>
    h             <sepcol> posizione attuale        <rowsep>
    t             <sepcol> superiore                <rowsep>
    b             <sepcol> inferiore                <rowsep>
    p             <sepcol> pagina                   <rowsep>
    <hline>
</tabular>
<caption>
        <label id="tabella-esempio">
        Esempio di tabella.
</caption>
</table>
```

### 53.1.2.12 Mappa dei caratteri

Alcuni caratteri che all'interno di LinuxDoc hanno un significato speciale, oltre a quelli che sono al di fuori della codifica ASCII standard, possono essere inseriti nel testo finale utilizzando dei codici macro; precisamente si tratta delle entità standard.[4]

Questi codici macro sono preceduti dalla e-commerciale ('**&**') e seguiti da un punto e virgola. Nel capitolo 51 appare una tabella riferita alle entità standard di uso comune nell'SGML. Si tratta precisamente della tabella 51.25.

### 53.1.3 Struttura di DebianDoc

La struttura di un sorgente SGML secondo il DTD DebianDoc ricalca quello che si può vedere dall'esempio seguente:

```
<!DOCTYPE debiandoc PUBLIC "-//DebianDoc//DTD DebianDoc//EN">
<debiandoc>
    <book>
        <titlepag>
            <title>Titolo del documento</title>
            <author>
                <name>Pinco Pallino</name>
                <email>ppallino@dinkel.brot.dg</email>
            </author>
            <version>29/02/1999</version>
            <abstract>
                Breve introduzione al documento.
            </abstract>
            <copyright>
                <copyrightsummary>
                    Copyright &copy; 1999 Pinco Pallino
                </copyrightsummary>

                <p>This work is free; you can redistribute it
                and/or modify it under the terms of the GNU
                General Public License as published by the
                Free Software Foundation; either version 2 of
                the License, or (at your option) any later
                version.</p>

            </copyright>
        </titlepag>

        <toc detail="sect">

        <chapt id="primo-capitolo">
            <heading>Primo capitolo</heading>

            <p>Contenuto del primo capitolo,
            ...
            ...
            </p>

            <sect id="prima-sezione">
                <heading>Prima sezione del primo
                capitolo</heading>

                <p>Contenuto della prima sezione,
                ...
                ...
                </p>

            </sect>
            ...
            ...
        </chapt>
        ...
        ...
        <appendix id=prima-appendice>
                <heading>Prima appendice</heading>

                <p>...
                ...
                </p>

        </appendix>
        ...
        ...
    </book>

</debiandoc>
```

Si può osservare una grande affinità con il DTD LinuxDoc, dove spicca in particolare il fatto che le etichette per la realizzazione di riferimenti incrociati sono inserite come attributi '**ID**' degli elementi di suddivisione del testo: '**chapt**', '**sect**',...

DebianDoc presume quindi che si tratti di un libro suddiviso in capitoli, gli elementi '**chapt**', quindi in sezioni a vari livelli: '**sect**', '**sect1**', '**sect2**', '**sect3**' e '**sect4**'.

È speciale anche l'elemento di dichiarazione dell'indice generale, '**toc**', che prevede l'attributo '**DETAIL**', al quale si deve assegnare il nome del livello di suddivisione che si ritiene indispensabile includere nell'indice generale: nell'esempio mostrato vengono inclusi solo i capitoli e le sezioni del livello iniziale.



#### 53.1.3.1 Organizzazione del catalogo, del DTD e delle entità

Dal punto di vista dell'SGML, DebianDoc è organizzato con un catalogo unico, che contiene le indicazioni seguenti:

```
DOCTYPE debiandoc                          dtd/debiandoc.dtd
PUBLIC "-//DebianDoc//DTD DebianDoc//EN"   dtd/debiandoc.dtd
ENTITY %general-chars                      entities/general
```

Queste righe vengono aggiunte al catalogo del sistema, corrispondente a '`/usr/share/sgml/catalog`', che in pratica è un collegamento simbolico al file '`/etc/sgml.catalog`'. Leggendo le dichiarazioni del catalogo si intende che il DTD DebianDoc è costituito dal file '`dtd/debiandoc.dtd`', ovvero '`/usr/share/sgml/dtd/debiandoc.dtd`'; inoltre, si vede che viene usato un solo file di entità generali: '`entities/general`', ovvero '`/usr/share/sgml/entities/general`'.

#### 53.1.3.2 Utilizzo sommario

Attraverso gli strumenti di DebianDoc, si ottiene un documento finale a partire da un sorgente SGML. Per questo, si elabora il sorgente come si fa con un linguaggio di programmazione durante la compilazione.

```
debiandoc2dvi [-k] [-p formato_carta] file_sgml
```

```
debiandoc2dvips [-k] [-p formato_carta] file_sgml
```

```
debiandoc2html [-k] file_sgml
```

```
debiandoc2info [-k] file_sgml
```

```
debiandoc2latex2e [-k] [-0] [--] file_sgml
```

```
debiandoc2lout [-k] [-0] [--] file_sgml
```

```
debiandoc2ps [-k] [-0] [-1] [-p formato_carta] [--] file_sgml
```

```
debiandoc2texinfo [-k] [-0] [--] file_sgml
```

```
debiandoc2text [-k] [-0] [--] file_sgml
```

```
debiandoc2textov [-k] [-0] [--] file_sgml
```

Ognuno di questi comandi elencati rappresenta un modo differente di elaborare e convertire un sorgente SGML scritto secondo il DTD DebianDoc. Il significato dei nomi dovrebbe essere intuitivo: '**debiandoc2html**' significa evidentemente «DebianDoc to HTML», ovvero, «da DebianDoc a HTML». Lo stesso vale, più o meno, per gli altri comandi. In breve:

- '**debiandoc2latex2e**' produce un file LaTeX;
- '**debiandoc2dvi**' produce un file DVI attraverso l'elaborazione con il sistema di composizione LaTeX;
- '**debiandoc2dvips**' produce un file PostScript attraverso l'elaborazione con il sistema di composizione LaTeX;

- '**debiandoc2html**' produce una trasformazione in HTML, distribuita su più file con estensione '`.html`', collocati in una directory il cui nome corrisponde alla radice del file sorgente;
- '**debiandoc2texinfo**' produce un file in formato Texinfo;
- '**debiandoc2info**' produce un file di documentazione Info, attraverso il sistema di composizione Texinfo;
- '**debiandoc2lout**' produce un file adatto per il sistema di composizione Lout;
- '**debiandoc2ps**' produce un file PostScript, attraverso l'elaborazione del sistema di composizione Lout, in cui le pagine sono ridotte e raddoppiate (ogni pagina A4 ne contiene due A5, a meno che venga utilizzata l'opzione '**-1**');
- '**debiandoc2text**' produce un file di testo puro e semplice, con un'ampiezza di 79 colonne;
- '**debiandoc2textov**' produce un file di testo con i codici di arretramento per ottenere gli effetti di evidenziamento e sottolineatura per la visualizzazione su schermo.

| Opzione | Descrizione |
|---|---|
| `-k` | Fa in modo che i file intermedi, creati durante il procedimento di conversione, vengano conservati. |
| `-O` | Fa in modo che il risultato finale della trasformazione venga emesso attraverso lo standard output, quando di solito si crea invece un file con la stessa radice dell'origine e un'estensione opportuna. Se il sorgente è fornito attraverso lo standard input, questa opzione è implicita. |
| `-1` | Questa opzione riguarda espressamente '**debiandoc2ps**', che senza di questa, genera un file PostScript in cui ogni pagina ne contiene due ridotte e affiancate (per mezzo di PSUtils). Con questa opzione, si ottengono pagine normali (singole). |
| `-p` ***dimensione_pagina*** | Questa opzione permette di specificare la dimensione della pagina, nelle trasformazioni in cui ciò può avere senso, facendo riferimento alla configurazione del pacchetto Papersize della distribuzione Debian. |
| `--` | In caso di ambiguità, un trattino doppio serve a separare le opzioni dal nome del file sorgente. |

### 53.1.4 Guida rapida di DebianDoc

Dal momento che DebianDoc è molto simile a LinuxDoc e che la sua documentazione è abbastanza chiara, non è il caso di ripetere le stesse informazioni anche in questo capitolo. Eventualmente si può rileggere quello precedente. Qui vengono mostrati solo i prospetti riassuntivi degli elementi SGML principali di DebianDoc, attraverso delle tabelle.

Tabella 53.37. Elementi della struttura generale di un documento DebianDoc.

| Elemento | Descrizione |
|---|---|
| `debiandoc` | Il contenitore di un documento DebianDoc. |
| `book` | Il sotto-contenitore di un documento DebianDoc. |
| `titlepag` | La definizione della pagina del titolo. |
| `title` | Il titolo del documento. |
| `author` | L'autore (scomposto ulteriormente). |
| `name` | Il nome dell'autore. |
| `email` | L'indirizzo di posta elettronica dell'autore. |



| Elemento | Descrizione |
|---|---|
| `version` | La versione del documento. |
| `abstract` | Una descrizione breve del contenuto. |
| `copyright` | Informazioni sul copyright. |
| `copyrightsummary` | Il copyright, in breve. |
| `p` | La descrizione della licenza. |
| `toc` | L'indice generale. |
| `chapt` | Il contenitore di un capitolo. |
| `appendix` | Il contenitore di un'appendice. |

Tabella 53.38. Elementi che rappresentano la suddivisione gerarchica del contenuto di un documento DebianDoc.

| Elemento | Descrizione |
|---|---|
| `chapt` | Il contenitore di un capitolo. |
| `appendix` | Il contenitore di un'appendice (si articola come il capitolo). |
| `sect` | Sezione di un capitolo o di un'appendice. |
| `sect1` | Sotto-sezione di primo livello. |
| `sect2` | Sotto-sezione di secondo livello. |
| `sect3` | Sotto-sezione di terzo livello. |
| `sect4` | Sotto-sezione di quarto livello. |
| `heading` | Il titolo di: capitolo, appendice, sezione o sotto-sezione. |

Tabella 53.39. Elementi che si utilizzano nel corpo del testo per modificare l'aspetto del loro contenuto in base al significato che rappresentano.

| Elemento | Descrizione |
|---|---|
| `em` | Enfasi normale (idealmente un corsivo). |
| `strong` | Enfasi più forte (idealmente un neretto). |
| `var` | Rappresentazione di una metavariabile (di uno schema sintattico). |
| `package` | Il nome di un pacchetto GNU/Linux. |
| `prgn` | Il nome di un programma o di un file ben conosciuto. |
| `file` | Il percorso di un file o di una directory. |
| `tt` | Una stringa letterale dattilografica. |

Tabella 53.40. Riferimenti.

| Elemento | Descrizione |
|---|---|
| `ref id="`***etichetta***`"` | Riferimento a un'etichetta dichiarata altrove. |
| `manref name="`***nome***`"` ↩ ↪`section="`***n_sezione***`"` | Riferimento a una pagina di manuale. |
| `email` | Contenitore di un indirizzo di posta elettronica. |
| `ftpsite` | Il nome a dominio di un sito FTP. |
| `ftppath` | Il percorso riferito all'ultimo sito FTP indicato. |
| `httpsite` | Il nome a dominio di un sito HTTP. |
| `httppath` | Il percorso riferito all'ultimo sito HTTP indicato. |
| `url id="`***uri***`" name="`***nome***`"` | Indirizzo URI completo. |
| `footnote` | Nota a piè pagina. |

Tabella 53.41. Elenchi.

| Elemento | Descrizione |
|---|---|
| `list` | Elenco puntato. |
| `item` | Voce di un elenco. |
| `enumlist` | Elenco numerato. |
| `item` | Voce di un elenco. |
| `taglist` | Elenco descrittivo. |
| `tag` | Elemento descrittivo. |
| `item` | Voce di un elemento. |

## 53.2 Introduzione a DocBook

DocBook è un formato per la composizione di documenti tecnici che ha l'intento di essere onnicomprensivo. Dal lato pratico, DocBook è essenzialmente un DTD, realizzato originariamente per l'SGML, successivamente adattato e ridotto a XML. Naturalmente, l'importanza di un DTD è minima se poi mancano degli strumenti di composizione adeguati. DocBook è molto complesso e gli strumenti realizzati attorno a lui sono diversi e in continua evoluzione.

In questo capitolo viene mostrato solo un utilizzo superficiale di strumenti per la composizione abbinati a DocBook, soprattutto in considerazione della dinamicità con cui queste cose si evolvono. Può essere molto utile la lettura di *DocBook demystification HOWTO*, di Eric S. Raymond, per avere una visione di insieme sulle strade che si possono scegliere per arrivare al risultato di proprio interesse.

### 53.2.1 Edizioni e varianti del DTD

L'importanza di DocBook è paragonabile a HTML; da questo fatto si può comprendere la presenza di diverse versioni e varianti di questo formato di composizione. Tra queste varianti possono essere interessanti anche quelle che cercano di ridurre il tutto a una struttura più semplice, pur rimanendo compatibile con quella generale complessiva, quasi come avviene con un formato HTML «stretto» (*strict*).

Se i vari DTD DocBook sono stati installati correttamente (dipende probabilmente dall'organizzazione della propria distribuzione GNU), dovrebbe essere sufficiente indicare l'intestazione corretta nel proprio sorgente per fare riferimento al DTD che si preferisce.

### 53.2.2 Esperimenti con il DTD e convalida

Per cominciare a fare qualche esperimento con il DTD DocBook, occorre almeno uno strumento di convalida, di solito il pacchetto SP di James Clark. Nella propria distribuzione GNU/Linux, questo pacchetto potrebbe essere disponibile da solo (come avviene nella distribuzione Debian), oppure assieme a Jade (come avviene nella distribuzione Red Hat). Quello che conta è, per iniziare, che sia disponibile l'eseguibile '**`nsgmls`**'.

Senza entrare nel dettaglio dell'SGML di DocBook, si può prendere l'esempio seguente come base per gli esperimenti.

```
<!DOCTYPE book PUBLIC "-//OASIS//DTD DocBook XML V4.2//EN"
   "http://www.oasis-open.org/docbook/xml/4.2/docbookx.dtd">
<book id="book" lang="it">
<bookinfo>
  <title>Il mio primo libro con DocBook</title>
  <authorgroup>
    <author>
      <surname>Pallino</surname>
      <firstname>Pinco</firstname>
    </author>
  </authorgroup>
  <editor>
    <surname>Cai</surname>
    <firstname>Caio</firstname>
  </editor>
  <legalnotice>
    <para>Copyright &copy; 1999 Pinco Pallino</para>
    <para>This work is free; you can redistribute it and/or
    modify it under the terms of the GNU General Public
    License as published by the Free Software Foundation;
    either version 2 of the License, or (at your option) any
    later version.</para>
  </legalnotice>
</bookinfo>

<chapter>
<title>Primo capitolo</title>

<para>Contenuto del primo capitolo, bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla.</para>

<para>bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla.</para>

<sect1>
<title>Prima sezione del primo capitolo</title>

<para>Contenuto della prima sezione, bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla.</para>

<para>bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla.</para>

</sect1>
</chapter>
<appendix>
<title>Prima appendice</title>

<para>Contenuto della prima appendice, bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla.</para>

<para>bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla.</para>

</appendix>

</book>
```

La verifica si fa nel modo già visto tante altre volte. Supponendo di voler fare riferimento al catalogo contenuto nel file '`/usr/share/sgml/catalog`' e supponendo di avere chiamato il sorgente SGML '`libro.sgml`':

```
$ nsgmls -s -c /usr/share/sgml/catalog libro.sgml [Invio]
```

Eventualmente, se il pacchetto di programmi che contiene SP è stato compilato in modo coerente con l'impostazione SGML della propria distribuzione, potrebbe non essere necessario indicare espressamente il file del catalogo:

```
$ nsgmls -s libro.sgml [Invio]
```

A questo punto, disponendo di un analizzatore SGML che funziona correttamente con questo DTD, si potrebbero realizzare i propri strumenti per la trasformazione in un risultato adatto alla consultazione: cartacea o elettronica.

Purtroppo può capitare di disporre di DTD DocBook che non sono stati realizzati in modo accurato. In questi casi si possono osservare una serie di segnalazioni di errori che dipendono da file estranei al proprio. Dal momento che '**nsgmls**' ha un limite massimo di errori predefinito, dopo il quale non ne mostra altri, potrebbe essere necessario utilizzare l'opzione '**-E**' per estenderlo, andando poi a cercare gli errori riferiti direttamente al file di proprio interesse:

```
$ nsgmls -E1000 -s libro.sgml [ Invio ]
```

### 53.2.3 Strumenti per la composizione

All'inizio del capitolo si è accennato al fatto che gli strumenti di composizione che si avvalgono di DocBook sono diversi, ma forse nessuno può dirsi abbastanza maturo per poter essere scelto senza alcun dubbio. In condizioni normali, senza entrare nel dettaglio di questi strumenti, potrebbero essere disponibili degli script che facilitano la produzione di un certo formato finale prescelto. Potrebbe trattarsi di nomi come quelli seguenti:

```
db2ps file_sgml_docbook | docbook2ps file_sgml_docbook
```

```
db2pdf file_sgml_docbook | docbook2pdf file_sgml_docbook
```

```
db2rtf file_sgml_docbook | docbook2rtf file_sgml_docbook
```

```
db2html file_sgml_docbook | docbook2html file_sgml_docbook
```

In generale, quindi, uno script del tipo '**db2***nome*' o '**docbook2***nome*' sta a indicare una conversione da DocBook al formato indicato dal ***nome***. In pratica, lo script va letto mnemonicamente come: *from docbook to* ***x***.

### 53.2.4 Struttura generale

Un documento DocBook si può articolare in modi differenti. Per prima cosa si distingue il tipo di documento; i casi principali si possono ridurre a: raccolta, libro o articolo. Lo schema della struttura di un libro si può semplificare nel modo seguente:

```
<!DOCTYPE book PUBLIC "-//OASIS//DTD DocBook XML V4.2//EN"
  "http://www.oasis-open.org/docbook/xml/4.2/docbookx.dtd">
<book>
<bookinfo>
    …
</bookinfo>
[<part>]
    <chapter>
        …
    </chapter>
[<part>]
…
[<appendix>
    …
</appendix>]
[<glossary>
    …
</glossary>]
</book>
```

A loro volta, i capitoli e le appendici, possono suddividersi in sezioni:



```
        [<sect1>
            [<sect2>
                [<sect3>
                    [<sect4>
                        [<sect5>
                            …
                        </sect5>]
                        …
                    </sect4>]
                    …
                </sect3>]
                …
            </sect2>]
            …
        </sect1>]
        …
```

La struttura che appare è incompleta e serve solo per avere un'idea dell'articolazione di un libro secondo DocBook. Si può osservare che a differenza di HTML, le sezioni di un documento (parti, capitoli e sezioni di livello inferiore) sono ben delimitate attraverso un elemento appropriato.

Una raccolta, ovvero un documento molto grande diviso in volumi, può essere strutturato nel modo seguente, dove naturalmente ogni elemento '**book**' si può articolare come già visto a proposito della struttura a volume singolo:

```
<!DOCTYPE set PUBLIC "-//OASIS//DTD DocBook XML V4.2//EN"
  "http://www.oasis-open.org/docbook/xml/4.2/docbookx.dtd">
<set>
<setinfo>
    …
</setinfo>
<book>
    …
<book>
…
</set>
```

Un articolo, ovvero un documento di piccole dimensioni, può essere strutturato nel modo seguente:

```
<!DOCTYPE article PUBLIC
  "-//OASIS//DTD DocBook XML V4.2//EN"
  "http://www.oasis-open.org/docbook/xml/4.2/docbookx.dtd">
<article>
<articleinfo>
    …
</articleinfo>
{<sect1>
    {<sect2>
        {<sect3>
            {<sect4>
                {<sect5>
                    …
                </sect5>}
                …
            </sect4>}
            …
        </sect3>}
        …
    </sect2>}
    …
</sect1>}
…
</article>
```

Gli attributi più importanti che si possono usare con gli elementi che delimitano porzioni di testo, specie se si tratta di quelli che lo dividono in sezioni (parti, capitoli, ecc.), sono '**id**' e '**lang**'. Il primo serve a indicare una stringa di identificazione, che in seguito può essere usata per fare un riferimento incrociato; il secondo stabilisce il linguaggio, attraverso una sigla espressa secondo lo standard ISO 639 (tabella 13.4), cosa che può riflettersi in vario modo in fase di composizione.

Nei modelli mostrati, riferiti rispettivamente al libro, alla raccolta e all'articolo, appare inizialmente un elemento destinato all'inserimento di informazioni particolari: '**bookinfo**', '**setinfo**' e '**articolinfo**'. Questi due elementi sono paragonabili all'intestazione di un file HTML, dove va collocato il titolo, i dati dell'autore o degli autori, assieme a tutte le altre informazioni generali riferite all'opera. Il modello seguente semplifica questa struttura:

```
<bookinfo>|<setinfo>|<articleinfo>
    <title>titolo</title>
    <authorgroup>
        <author>
            <firstname>nome_autore</firstname>
            <familyname>cognome_autore</familyname>
        </author>
        …
    </authorgroup>
    <keywordset>
        <keyword>parola_chiave</keyword>
        …
    </keywordset>
    <abstract>
        descrizione_del_documento
    </abstract>
    <legalnotice>
        note_legali_sul_documento
    </legalnotice>
    <edition>dati_dell'edizione</edition>
</bookinfo>|</setinfo>|</articleinfo>
```

### 53.2.5 Corpo del documento

L'elemento più comune che appare nel corpo del documento e anche in altri contesti che lo richiedono è il paragrafo:

```
<para>testo</para>
```

Così come avviene nella maggior parte dei sistemi SGML (o XML), il paragrafo rappresenta un blocco di testo elementare, che contiene può contenere altri elementi che però non costituiscono un altro blocco separato.

Nelle sezioni seguenti vengono descritti altri elementi comuni del corpo del documento, senza però entrare nel dettaglio delle caratteristiche e delle possibilità di questi.

#### 53.2.5.1 Elenchi

DocBook dispone di una discreta quantità di elenchi differenti. L'elenco più semplice è definito dall'elemento '**simplelist**', allo scopo di annotare una serie di voci, senza un'estetica o uno scopo particolare:

```
<simplelist>
<member>prima_voce_dell'elenco</member>
<member>seconda_voce_dell'elenco</member>
…
<member>ultima_voce_dell'elenco</member>
</simplelist>
```

L'elenco puntato classico si ottiene con l'elemento '**itemizedlist**':



```
<itemizedlist>
<listitem>blocchi_della_prima_voce</listitem>
<listitem>blocchi_della_seconda_voce</listitem>
…
<listitem>blocchi_dell'ultima_voce</listitem>
</itemizedlist>
```

Si osservi che la differenza più importante tra i due tipi di elenchi appena descritti sta nel fatto che le voci contenute negli elementi '**listitem**' sono costituite da uno o più blocchi di testo (per esempio i paragrafi), mentre l'elemento '**member**' contiene un testo lineare che l'elemento stesso traduce in un blocco unico.

L'elenco numerato si ottiene con l'elemento '**orderedlist**':

```
<orderedlist>
<listitem>blocchi_della_prima_voce</listitem>
<listitem>blocchi_della_seconda_voce</listitem>
…
<listitem>blocchi_dell'ultima_voce</listitem>
</orderedlist>
```

Come si vede, '**orderedlist**' si utilizza nello stesso modo dell'elenco puntato normale, con la differenza che le voci sono numerate.

Gli elenchi descrittivi si ottengono con l'elemento '**variablelist**' che appare un po' complesso rispetto a quelli già presentati:

```
<variablelist>
<varlistentry>
    <term>voce_descrittiva</term>
    …
    <listitem>blocchi_della_voce</listitem>
</varlistentry>
…
</variablelist>
```

Lo schema mostra l'elemento '**variablelist**' che può contenere uno o più elementi '**varlistentry**', all'interno dei quali, ogni voce contiene una o più descrizioni all'interno di elementi '**term**', seguiti da un solo elemento '**listitem**' che ha le caratteristiche già viste a proposito degli elenchi puntati e numerati.

Quelli descritti sono elenchi classici che di solito si trovano nella maggior parte dei sistemi di composizione; tuttavia DocBook dispone anche di altri elenchi speciali, che qui non vengono descritti.

#### 53.2.5.2 Tabelle

DocBook offre tabelle molto complesse, la cui realizzazione pratica dipende però dagli strumenti di composizione utilizzati effettivamente. In generale si distinguono due involucri alternativi: l'elemento '**table**' e l'elemento '**informaltable**'. Nel primo caso si ha una tabella che richiede l'inserimento di un titolo (ovvero di una didascalia); nel secondo questo viene a mancare:

```
<table>
    <title>titolo</title>
    <tgroup cols="n_colonne">
        corpo_della_tabella
    </tgroup>
</table>
```

```
<informaltable>
    <tgroup cols="n_colonne">
        corpo_della_tabella
    </tgroup>
</informaltable>
```

Il contenuto dell'elemento '**tgroup**' è il corpo della tabella, con o senza intestazioni:

```
<tgroup cols="n_colonne">
    [<thead>
        righe
    </thead>]
    [<tfoot>
        righe
    </tfoot>]
    <tbody>
        righe
    </tbody>
</tgroup>
```

Il contenuto di un'intestazione o del corpo sono le righe, normalmente descritte semplicemente con l'elemento '**row**':

```
<row>
    <entry>
        contenuto_della_cella
    </entry>
    …
</row>
```

Si osservi che il contenuto di una cella può essere un testo lineare oppure uno o più blocchi.

### 53.2.5.3 Immagini

L'inserzione di immagini può avvenire in contesti differenti, utilizzando elementi appropriati che attribuiscono un senso al tipo di immagine che si va a includere. Il modo più generalizzato per inserire un'immagine è attraverso l'uso dell'elemento '**mediaobject**' o di '**inlinemediaobject**': il primo per le immagini che costituiscono un blocco a parte, mentre il secondo consente l'inserzione all'interno di un testo lineare. Inoltre, quando si inserisce un'immagine che costituisce un blocco autonomo, questa la si può inserire nell'elemento '**figure**' o nell'elemento '**informalfigure**'; entrambi consentono di controllarne la fluttuazione per motivi tipografici, mentre solo il primo permette di aggiungere un titolo alla figura.

```
<informalfigure [float="0"|"0"]>
    figura
    …
</informalfigure>
```

```
<figure [float="0"|"0"]>
    [<title>titolo</title>]
    figura
    …
</figure>
```

I due modelli sintattici riepilogano in modo molto semplificato l'uso degli involucri costituiti dagli elementi '**informalfigure**' e '**figure**'. Come si può intuire, assegnando il valore uno all'attributo '**float**' si rende fluttuante la figura contenuta.

Anche gli elementi '**mediaobject**' o di '**inlinemediaobject**' sono contenitori di altri più specifici per poter fare riferimento a un file esterno contenente un'immagine. Come accennato, il primo si presta all'uso in un blocco a sé stante, mentre il secondo per l'inserimento in un testo lineare. Nel caso si utilizzi un involucro del tipo '**figure**' o '**informalfigure**', si deve utilizzare '**mediaobject**'.



```
<inlinemediaobject>|<mediaobject>
    <imageobject>
        <imagedata fileref="file_esterno" format="nome_formato">
    </imageobject>
    …
    [<textobject>
        blocchi_di_testo
    </textobject>]
</inlinemediaobject>|</mediaobject>
```

Il modello sintattico descrive in modo molto semplificato l'uso degli elementi '**inlinemediaobject**' e '**mediaobject**', tralasciando la possibilità per il secondo di inserire una didascalia nell'elemento '**caption**'.[5]

Come si può vedere dal modello, si possono inserire più elementi '**imageobject**', ognuno per indicare il riferimento a un'immagine in forma diversa, da usare in alternativa alle altre, secondo le caratteristiche del tipo di composizione attuato. In questa ottica va visto anche l'elemento '**textobject**' che serve a contenere dei blocchi di testo normali, per descrivere l'oggetto quando questo non può essere visualizzato in alcun modo. L'esempio seguente mostra l'inserimento di un'immagine, senza l'involucro '**figure**', in cui appare il riferimento a due formati differenti, ognuno adatto a un sistema di composizione particolare, assieme a un'annotazione alternativa:

```
<mediaobject>
    <imageobject>
        <imagedata fileref="prova.jpg" format="jpg"
    </imageobject>
    <imageobject>
        <imagedata fileref="prova.eps" format="eps"
    </imageobject>
    <textobject>
        <para>una figura qualunque</para>
    </textobject>
    <caption>
        <para>Questa &egrave; una prova</para>
    </caption>
</mediaobject>
```

### 53.2.5.4 Riferimenti interni ed esterni

I riferimenti interni al documento si fanno inserendo delle ancore per mezzo dell'attributo '**id**' disponibile nella maggior parte degli elementi:

```
<nome_elemento id="stringa_di_identificazione">…</nome_elemento>
```

Eventualmente è disponibile anche un elemento aggiuntivo il cui solo scopo è quello di consentire l'inserimento di un'ancora in quel punto particolare; si tratta dell'elemento '**anchor**' che è vuoto:

```
<anchor id="stringa_di_identificazione"/>
```

Per fare riferimento alle ancore inserite nel documento stesso, si possono usare due modi, attraverso gli elementi '**link**' e '**xref**'. Nel primo caso si ottiene un riferimento ipertestuale attraverso il testo delimitato dall'elemento '**link**', valido quindi solo in una composizione che consenta questo tipo di utilizzo, mentre l'elemento '**xref**' è vuoto e fa apparire il numero della sezione o un altro riferimento visivo:

```
<link linkend="stringa_di_identificazione">…</link>
```

```
<xref linkend="stringa_di_identificazione"/>
```

I riferimenti esterni si fanno normalmente attraverso degli indirizzi URI; per questi è a disposizione l'elemento '**ulink**' che si usa con l'attributo '**url**':

```
<ulink url="indirizzo_uri">testo_del_riferimento</ulink>
```

A seconda della composizione finale, l'indirizzo URI a cui si fa riferimento potrebbe apparire oppure no.

### 53.2.6 Conclusione

DocBook è un sistema SGML/XML molto complesso e ancora lontano dall'avere trovato una struttura definitiva. Data la disponibilità di una quantità così grande di elementi, che spesso si equivalgono, oltre alla grande libertà con cui questi possono essere usati, fa sì che l'uso effettivo di DocBook dipenda principalmente da due fattori: le capacità reali del sistema di composizione e dalla guida di stile stabilita per il progetto editoriale a cui si vuole partecipare con DocBook.

In altri termini, la struttura di DocBook non dà implicitamente dei limiti e difficilmente un autore può conoscere perfettamente il contesto corretto per ogni elemento; pertanto, diventa indispensabile la definizione di una guida di stile per l'uso di DocBook; guida che può essere molto diversa da un progetto editoriale a un altro.

## 53.3 Introduzione a TEI

TEI (*Text encoding initiative*), è un sistema di codifica SGML/XML per testi, nato e sviluppato con l'intento di dare la possibilità di trascrivere in forma elettronica, un documento che originariamente è disponibile solo in forma cartacea.

A prima vista non si comprende il significato di questa precisazione, se non si considera lo studio di un testo dal punto di vista linguistico, storico e artistico. Tanto per fare un esempio, TEI prevede anche la possibilità di annotare dove si trovano i salti pagina e le interruzioni di riga, nell'ambito di una certa edizione del documento che viene trascritto.

In generale, TEI potrebbe essere usato per realizzare qualunque tipo di documento elettronico, ma la sua caratteristica lo porta a essere più adatto alla trascrizione di ciò che non nasce in forma elettronica.

Il DTD di TEI prevede una grande quantità di elementi, con scopi molto particolari. Sta comunque a chi lo utilizza (per scrivere un'opera originale o per trascrivere un altro documento), stabilire come analizzare e suddividere le informazioni, in base all'obiettivo che si prefigge con il suo lavoro.

TEI offre anche un sottoinsieme del DTD completo, denominato TEI Lite. In generale i sistemi di composizione basati su TEI potrebbero non essere in grado di gestire tutte le situazioni, pertanto conviene avvicinarsi a questo sistema iniziando dal modello semplificato.

Lo sviluppo di TEI è sostenuto da associazioni letterarie, linguistiche e umanistiche che hanno dato vita a un consorzio: il consorzio TEI.

Le sezioni successive offrono solo una panoramica superficiale delle caratteristiche di TEI, assieme a qualche piccolo esempio pratico. La documentazione citata alla fine del capitolo può servire per approfondirne lo studio.

### 53.3.1 Strumenti per la composizione

In un sistema GNU esistono due modi per comporre un documento scritto secondo il formato XML di TEI: attraverso dei fogli di stile XSLT che permettono di ottenere un file XSL-FO da comporre successivamente, oppure attraverso PassiveTeX (e di conseguenza XMLTeX). In generale, nessuno dei due modi dà risultati perfetti, pertanto conviene considerarli entrambi.

```
<?xml version="1.0" encoding="UTF-8" ?>
<TEI.2>
    <teiHeader>
        <fileDesc>
            <titleStmt>
                <title>Esempio elementare con TEI</title>
            </titleStmt>
            <publicationStmt>
                <p/>
            </publicationStmt>
            <sourceDesc>
                <p/>
            </sourceDesc>
        </fileDesc>
    </teiHeader>
    <text>
        <body>
            <p>Ciao a tutti!</p>
        </body>
    </text>
</TEI.2>
```

L'esempio che si vede rappresenta il minimo necessario per scrivere una piccola frase. Supponendo di avere chiamato questo file '`tei-001.xml`', dovrebbe essere possibile ottenere la sua composizione con l'aiuto di PassiveTeX e XMLTeX in questo modo:

```
$ latex "&xmltex" tei-001.xml [Invio]
```

Dal momento che non ci sono riferimenti incrociati, non serve ripetere il procedimento. Eventualmente, si può convertire il file DVI ottenuto in PostScript:

```
$ dvips -o tei-001.ps tei-001.dvi [Invio]
```

In alternativa a questi passaggi si potrebbe generare direttamente un file PDF in questo modo:

```
$ pdflatex "&pdfxmltex" tei-001.xml [Invio]
```

Se invece si vuole percorrere la strada della trasformazione XSLT e della composizione del file XSL-FO che si genera, occorre procurarsi i fogli di stile, dall'indirizzo *http://sourceforge.net/projects/tei/files/* .

Una volta ottenuti tutti i file che compongono il pacchetto dei fogli di stile per la composizione per la stampa, si può procedere alla trasformazione con strumenti come Xalan (sezione 52.2):

```
$ xalan -IN tei-001.xml -XSL tei.xsl -OUT tei-001.fo [Invio]
```

Si dovrebbe ottenere così il file '`tei-001.fo`' da rielaborare ulteriormente.

Il lavoro di realizzazione dei fogli di stile XSLT potrebbe essere incompleto, tanto da far sì che Xalan riveli degli errori. Durante le prove fatte per realizzare questo capitolo il file '`teicommon.xsl`', del pacchetto già citato, conteneva un errore relativo alla chiamata di un modello non dichiarato (precisamente si trattava di '**`makeURL`**'). È stato sufficiente circoscrivere con un commento l'elemento '**`xsl:call-template`**' che provocava l'errore per completare la trasformazione in modo corretto.

La composizione del file XSL-FO generato si può ottenere per mezzo di FOP, oppure sempre con PassiveTeX e XMLTeX. Qui si mostra il secondo caso; per l'uso di FOP si può consultare la sezione 52.3:

```
$ pdflatex "&pdfxmltex" tei-001.fo [Invio]
```

### 53.3.2 Struttura generale di un documento TEI

Un documento TEI è contenuto nell'elemento '**`TEI.2`**', che si compone necessariamente degli elementi '**`teiHeader`**' e '**`text`**'. A sua volta, l'elemento '**`teiHeader`**' si scompone in altri elementi, parte dei quali sono obbligatori. Il modello seguente si riferisce al minimo della struttura complessiva:

```
<TEI.2>
    <teiHeader>
        <fileDesc>
            <titleStmt>
                <title>titolo</title>
                [<author>autore</author>]
                …
                [<editor>editore</editor>]
                …
            </titleStmt>
            <publicationStmt>
                dichiarazione_relativa_alla_pubblicazione
            </publicationStmt>
            <sourceDesc>
                descrizione_della_fonte
            </sourceDesc>
        </fileDesc>
    </teiHeader>
    <text>
        <body>
            corpo
        </body>
    </text>
</TEI.2>
```

Il contenuto del documento, circoscritto dall'elemento '**text**', può contenere, oltre all'elemento '**body**', gli elementi '**front**' e '**back**', per inserire rispettivamente, qualcosa da mettere all'inizio (come una prefazione) e alla fine (come un'appendice o un indice analitico). Al posto dell'elemento '**body**' si può mettere un elemento '**group**', che a sua volta può contenere un elemento '**text**' con la sua suddivisione. Queste possibilità vengono solo accennate, ma non si mostrano esempi di questo tipo.

### 53.3.2.1 Suddivisione del testo

L'elemento '**body**' (così come '**front**' e '**back**') può contenere del testo senza suddivisioni, composte per esempio da elementi '**p**' (paragrafi, come per HTML), oppure una suddivisione in sezioni, ma forse è più appropriato definirle «divisioni», delineate da elementi '**div**' o '**div*n***':

```
<div>
    …
    <div>
        …
        <div>
            …
            …
        </div>
    </div>
</div>
```

```
<div1>
    …
    <div2>
        …
        <div3>
            …
            …
        </div3>
    </div2>
</div1>
```

I due modelli sintattici dovrebbero permettere di capire che esiste un modo di usare gli elementi '**div**', annidandoli, lasciando intendere il livello in base all'annidamento, mentre è anche possibile usare elementi '**div*n***' che devono essere collocati in modo appropriato in base all'annidamento. In pratica, il secondo modo consente di avere un controllo maggiore rispetto agli errori che si possono fare involontariamente. È anche evidente che le due forme di suddivisione non sono miscelabili.



I vari attributi di questi elementi consentono di inserire informazioni che possono tornare utili per descrivere le caratteristiche della divisione, ma in linea di principio non intervengono nel modificare l'aspetto della composizione finale.

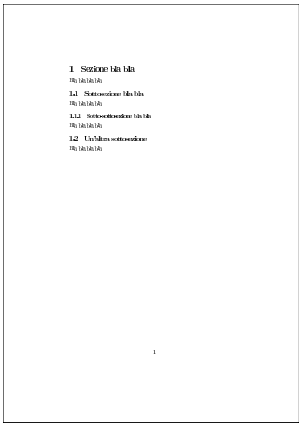
Una divisione ha probabilmente un titolo, che si definisce utilizzando l'elemento '**head**' prima del testo che si vuole inserire al suo interno:

```
<div[n]>
    <head>titolo</head>
    contenuto
    [<div[n]>
        …
    </div[n]>]
</div[n]>
```

Segue un esempio molto semplice, con l'uso di divisioni non numerate:

```
<?xml version="1.0" encoding="UTF-8" ?>
<TEI.2>
    <teiHeader>
        <fileDesc>
            <titleStmt>
                <title>Esempio elementare con TEI</title>
            </titleStmt>
            <publicationStmt>
                <p/>
            </publicationStmt>
            <sourceDesc>
                <p/>
            </sourceDesc>
        </fileDesc>
    </teiHeader>
    <text>
        <body>
            <div>
                <head>Sezione bla bla</head>
                <p>Bla bla bla bla</p>
                <div>
                    <head>Sottosezione bla bla</head>
                    <p>Bla bla bla bla</p>
                    <div>
                        <head>Sotto-sottosezione bla
                              bla</head>
                        <p>Bla bla bla bla</p>
                    </div>
                </div>
                <div>
                    <head>Un'altra sottosezione</head>
                    <p>Bla bla bla bla</p>
                </div>
            </div>
        </body>
    </text>
</TEI.2>
```

Si può osservare il risultato della composizione ottenuta con PassiveTeX e XMLTeX nella figura successiva:




### 53.3.2.2 Blocchi comuni

Dagli esempi mostrati è già apparso l'uso dell'elemento '**p**', che idealmente rappresenta ciò che viene chiamato normalmente «paragrafo», allo scopo di racchiudere del testo lineare, trasformandolo così in un blocco impaginato.

In generale non c'è molto da aggiungere a proposito di questo elemento, se non avvisare che sono disponibili diversi attributi per qualificare il testo che contiene.

```
<p>testo_lineare</p>
```

Negli esempi già mostrati appare anche l'uso di paragrafi vuoti, rappresentati come '**<p/>**', quando il contesto richiede espressamente l'inserimento di un blocco anche quando non si vuole fornire alcuna indicazione.

### 53.3.3 Elenchi

Gli elenchi si realizzano con l'elemento '**list**', che contiene normalmente soltanto elementi '**item**'. Notoriamente si distinguono diversi tipi di elenchi, in base al fatto che siano puntati, numerati o descrittivi. Questa caratteristica, contrariamente ad altri sistemi SGML/XML si ottiene assegnando il termine appropriate all'attributo '**type**':

```
<list type="simple"|"bulleted"|"ordered">
    <item>testo_lineare|blocco</item>
    …
</list>
```

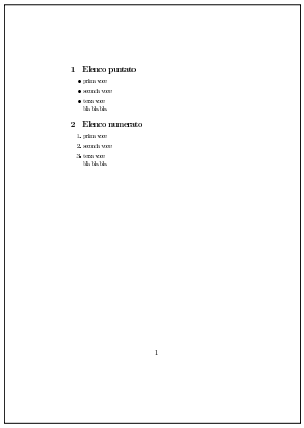
```
<list type="gloss">
    <label>etichetta</label> <item>testo_lineare|blocco</item>
    …
</list>
```

I due modelli sintattici sono semplificati rispetto alle possibilità, soprattutto per quanto riguarda la disponibilità di altri attributi non indispensabili. Dal confronto, si deve intendere che il secondo modello riguarda gli elenchi descrittivi, dove la componente descrittiva dei punti dell'elenco è contenuta nell'elemento '**label**'.

```
<div>
    <head>Elenco puntato</head>
    <list type="bulleted">
        <item>prima voce</item>
        <item>
            <p>seconda voce</p>
        </item>
        <item>
            <p>terza voce</p>
            <p>bla bla bla</p>
        </item>
    </list>
</div>
<div>
    <head>Elenco numerato</head>
    <list type="ordered">
        <item>prima voce</item>
        <item>
            <p>seconda voce</p>
        </item>
        <item>
            <p>terza voce</p>
            <p>bla bla bla</p>
        </item>
    </list>
</div>
```



L'elenco mostra due elenchi, uno puntato e l'altro numerato, contenuti ognuno in una divisione distinta. Si può osservare il fatto che gli elementi '**item**' possono contenere testo lineare o altri blocchi, pertanto si possono realizzare anche dei sottoelenchi. Il risultato della composizione ottenuta con PassiveTeX e XMLTeX appare nella figura successiva:




### 53.3.4 Tabelle

Le tabelle di TEI, pur essendo evidentemente dei blocchi, devono essere contenute all'interno di un elementi '**p**' o simili. Una tabella si compone essenzialmente secondo la sintassi seguente:

```
<table rows="n_righe" cols="n_colonne">
    <row>
        <cell>contenuto_cella</cell>
        …
    </row>
    …
</table>
```

Naturalmente sono disponibili molti attributi per gli elementi, oltre quelli che sono stati mostrati; inoltre, è possibile inserire anche altri tipi di elementi prima della descrizione delle righe, ma questo tipo di utilizzo non viene mostrato.

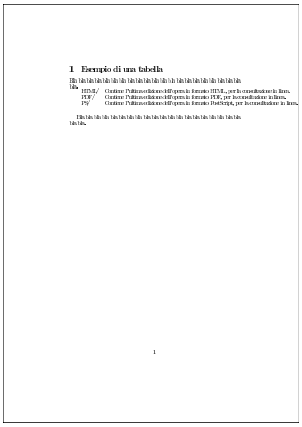
Le celle della tabella possono contenere sia testo lineare, sia blocchi; tuttavia, può darsi che il sistema di composizione non sia in grado di gestire celle che contengono testo molto lungo.

```
<div>
    <head>Esempio di una tabella</head>

    <p>Bla bla bla bla bla bla bla bla bla bla
    bla bla bla bla bla bla bla bla bla bla
    bla.</p>
    <p>
    <table rows="3" cols="2">
        <row>
            <cell>HTML/</cell>
            <cell>Contiene l'ultima edizione
            dell'opera in formato HTML, per la
            consultazione in linea.</cell>
        </row>
```

```
                    <row>
                        <cell>PDF/</cell>
                        <cell>Contiene l'ultima edizione
                        dell'opera in formato PDF, per la
                        consultazione in linea.</cell>
                    </row>
                    <row>
                        <cell>PS/</cell>
                        <cell>Contiene l'ultima edizione
                        dell'opera in formato PostScript,
                        per la consultazione in linea.</cell>
                    </row>
                </table>
                </p>
                <p>Bla bla bla bla bla bla bla bla bla bla
                bla bla bla bla bla bla bla bla bla bla bla
                bla.</p>
            </div>
```

L'esempio mostra una divisione che contiene una tabella molto semplice. Alcune celle della tabella contengono un testo molto lungo: dipende dal sistema di composizione effettivo la capacità o meno di impaginarlo correttamente. Nella figura successiva si vede cosa può accadere se ciò non avviene:



### 53.3.5 Figure

«

Anche le figure devono essere contenute all'interno di un elementi '**p**' o simile. Una figura si compone essenzialmente secondo la sintassi seguente:

```
<figure [entity="entità_generale"]>
    <figDesc>descrizione_opzionale</figDesc>
</figure>
```

Una figura, secondo il sistema TEI può anche non essere dichiarata come oggetto grafico, ma semplicemente come indicatore della sua presenza. Pertanto, l'attributo '**entity**' che ha lo scopo di fare riferimento al file che contiene l'immagine, è facoltativo. Si osservi che l'elemento '**figDesc**' serve a inserire una descrizione che permetta di dare qualche indicazione a chi consulta il documento senza la possibilità di visualizzare l'immagine; pertanto non è da confondere con una didascalia.

L'indicazione del file che contiene l'immagine da visualizzare non avviene nello stesso modo di altri sistemi SGML/XML, perché si fa riferimento al nome di un'entità generale che deve essere dichiarata all'inizio del documento. Si osservi l'esempio seguente, nel quale viene dichiarata l'entità '**FiguraEsempio**', contenente il nome del file da inserire ('`esempio.jpg`'), a cui si fa riferimento nell'elemento '**figure**':

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE TEI.2 SYSTEM "teixlite.dtd"[
<!ENTITY FiguraEsempio SYSTEM "esempio.jpg" NDATA jpeg>
]>
<TEI.2>
    <teiHeader>
```



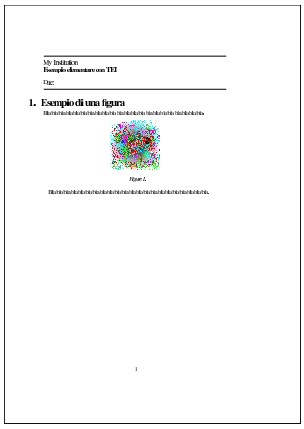
```
        <fileDesc>
            <titleStmt>
                <title>Esempio elementare con TEI</title>
            </titleStmt>
            <publicationStmt>
                <p/>
            </publicationStmt>
            <sourceDesc>
                <p/>
            </sourceDesc>
        </fileDesc>
    </teiHeader>
    <text>
        <body>
            <div>
                <head>Esempio di una figura</head>
                <p>Bla bla bla bla bla bla bla bla bla bla
                bla bla bla bla bla bla bla bla bla bla bla
                bla.</p>
                <p>
                <figure entity="FiguraEsempio">
                    <figDesc>Una figura di esempio senza
                    alcun significato.</figDesc>
                </figure>
                </p>
                <p>Bla bla bla bla bla bla bla bla bla bla
                bla bla bla bla bla bla bla bla bla bla bla
                bla.</p>
            </div>
        </body>
    </text>
</TEI.2>
```

Come si può osservare, nella dichiarazione dell'entità generale è stato specificato il formato dell'immagine. Sono disponibili le definizioni seguenti per i formati più comuni:

| Sigla | Formato corrispondente |
|---|---|
| '**cgm**' | CGM |
| '**png**' | PNG |
| '**tiff**' | TIFF |
| '**gif**' | GIF |
| '**jpeg**' | JPEG |

È ancora più importante osservare che il nome dell'entità viene inserito nell'attributo '**entity**' senza apparire come macro, ovvero senza la e-commerciale ('**&**') che servirebbe a espanderne il contenuto. Infatti, se così fosse, sarebbe sufficiente mettere il nome del file direttamente.

Si può vedere il risultato della composizione ottenuta attraverso la trasformazione dei fogli di stile XSLT nella figura seguente. Rispetto ad altri esempi che sono stati trasformati con PassiveTeX e XMLTeX, si notano delle informazioni in più.



### 53.3.6 Forme di evidenziamento

TEI prevede diverse forme di evidenziamento del testo, ma per «evidenziamento» si deve intendere qualcosa che va oltre il puro aspetto visivo della composizione finale, perché lo scopo è quello di descrivere ciò che era in origine. Per esempio, si può delimitare una porzione del testo specificando che si tratta di un corsivo, ma questo è importante soprattutto per sapere che così era nel testo originale che si trascrive, mentre non è sicuro che il sistema di composizione usato renda effettivamente in corsivo il testo.

Gli elementi destinati a contenere testo prevedono un attributo comune, denominato '**rend**' (*rendition*), il cui scopo prevalente è quello di descrivere l'aspetto del carattere tipografico. A questo attributo si possono associare nomi come '**italic**', '**bold**', '**roman**', '**gothic**' e tanti altri, ma non sempre il sistema di composizione è in grado di riconoscerli.

Tabella 53.54. Riepilogo di alcuni elementi e attributi utili per descrivere porzioni di testo con caratteristiche o significati particolari.

| Sintassi | Descrizione |
|---|---|
| `<`***nome*** `rend="`***definizione***`">` | L'attributo comune '**rend**' permette di dichiarare le caratteristiche tipografiche dell'oggetto delineato dall'elemento. |
| `<emph>`***testo***`</emph>` | Delinea un testo enfatizzato dal punto di vista linguistico; per stabilire l'aspetto del carattere tipografico si usa l'attributo '**rend**'. |
| `<hi>`***testo***`</hi>` | Delinea un testo enfatizzato dal punto di vista tipografico; per stabilire l'aspetto del carattere tipografico si usa l'attributo '**rend**'. |
| `<foreign lang="`***linguaggio***`">` ***testo*** `</foreign>` | Delinea un testo espresso in un'altra lingua. |
| `<term>`***testo***`</term>` | Delinea un termine tecnico. |
| `<title>`***testo***`</title>` | Delinea il titolo di qualcosa. |
| `<q>`***testo***`</q>` | Delinea una citazione. |

### 53.3.7 Note

TEI prevede diversi tipi di note (note a margine, note a piè pagina, ecc.), per poter descrivere sia le annotazioni presenti nel testo che si va a trascrivere, sia quelle che potrebbero essere aggiunte in fase di trascrizione. Queste annotazioni si inseriscono con l'elemento '**note**', che si utilizza in un contesto lineare:

```
<note   place="foot"|"inline"|"left"|"right"|"end"|…
        [resp="author"|"editor"|…]
        [altri_attributi]>
    nota
</note>
```

Come si vede, nel modello sintattico si vede anche la possibilità di usare l'attributo '**resp**', con lo scopo di specificare chi ha fatto la nota.

L'esempio seguente mostra la dichiarazione di una nota a piè pagina comune, senza indicazioni particolari:

```
        <p>Per esempio, /uno/due/tre rappresenta il file (o
        la directory) tre che discende da due, che discende
        da uno, che a sua volta discende dall'origine.<note
        place="foot">Il tipo di barra obliqua che si
        utilizza dipende dal sistema operativo. La barra
        obliqua normale corrisponde al sistema
        tradizionale.</note></p>
```



### 53.3.8 Riferimenti incrociati e riferimenti esterni

Quando si fanno riferimenti interi al documento, si utilizzano degli elementi che puntano a delle ancore, ovvero delle etichette, dichiarate attraverso un attributo comune alla maggior parte degli altri elementi: '**id**'. In mancanza di altro, si possono usare gli elementi '**anchor**' e '**seg**' per inserire un'etichetta:

```
<anchor id="etichetta" />
```

```
<seg id="etichetta">
    testo
</seg>
```

Per fare riferimento a queste ancore, si usano gli elementi '**ref**' e '**ptr**':

```
<ptr target="etichetta" />
```

```
<ref target="etichetta">
    testo
</ref>
```

Per fare riferimento a documenti esterni, le cose si complicano, perché occorre dichiarare gli indirizzi all'interno di entità generali, come avviene per i file delle figure:

```
<xptr doc="entità" />
```

```
<xref doc="entità">
    testo
</xref>
```

L'esempio seguente mostra l'uso di un riferimento interno e di uno esterno; alcune righe sono state omesse:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE TEI.2 SYSTEM "teixlite.dtd"[
<!ENTITY Indirizzo SYSTEM "http://www.brot.dg" NDATA URL>
]>
<TEI.2>
    ...
    ...
    <text>
        <body>
            <div>
                <head id="riferimenti">Riferimenti</head>
                ...
                ...
                <p>Per esempio, all'inizio della sezione
                <ref target="riferimenti">Riferimenti</ref>,
                si trova una spiegazione... bla bla bla bla
                bla bla bla bla bla bla bla.</p>
                ...
                ...
                <p>Il documento tal dei tali si ottiene
                dall'indirizzo <xref
                doc="Indirizzo">http://www.brot.dg</xref>.
                </p>
            </div>
        </body>
    </text>
</TEI.2>
```

### 53.3.9 Documentazione tecnica

Quando si scrive documentazione tecnica diventa importante poter rappresentare del testo con un carattere a larghezza uniforme, spesso

rispettando le interruzioni di riga. Per questo si usano due elementi in particolare: '**eg**' e '**code**':

```
<code>testo_lineare</code>
```

```
<eg>
    testo_lineare
    ...
</eg>
```

Dal modello sintattico potrebbe non essere chiaro, ma l'elemento '**eg**' è quello che rispetta le interruzioni di riga.

Naturalmente, dove appropriato, si possono usare sezioni marcate di tipo CDATA ('**<![[CDATA[***testo_letterale***]]>**'), per poter usare letteralmente alcuni simboli che per SGML e XML hanno significati particolari.

L'esempio seguente riepiloga l'uso di questi due elementi:

```
        <eg>
        dd              gzip            nisdomainname   tar
        df              hostname        ping            touch
        dmesg           kill            ps              true
        dnsdomainname   ln              pwd             umount
        doexec          login           rm              uname
        domainname      ls              rmdir           vi
        echo            mail            rpm             view
        egrep           mkdir           sed             vim
        ex              mknod           sh              zcat
        false           more            sleep
        </eg>

        <p>Il comando <code>mknod</code> crea...</p>
```

### 53.3.10 Indici

Come in altri sistemi di composizione SGML o XML, anche con TEI è possibile generare automaticamente degli indici. Si usa per questo l'elemento '**divGen**', che va a collocarsi al posto di un elemento '**div**' o '**div1**':

```
<divGen type="toc"/>
```

Quello che si vede è il modo per ottenere un indice generale dai titoli delle divisioni. Per ottenere un indice analitico occorre prima inserire degli elementi '**index**' nel testo, dove si possono individuare dei termini importanti da annotare nell'indice:

```
<index  level1="voce"
        [level2="voce_inferiore"
        [level3="voce_inferiore"
        [...]]] />
```

In pratica, è possibile indicare una voce singola, oppure una voce suddivisa in più livelli. Per ottenere l'indice analitico si usa sempre l'elemento '**divGen**', specificando che si tratta di un indice analitico:

```
<divGen type="index"/>
```

L'esempio seguente mostra i due casi. Si osservi che probabilmente gli strumenti di composizione comuni portano a ottenere l'indice generale, ma non quello analitico:

```
    <divGen type="toc" />
    <div>
        <head>Prima divisione</head>

        <p>I <index level1="transistor" />transistor si
        dividono in due tipi: <index level1="transistor"
        level2="PNP"/>PNP e <index level1="transistor"
        level2="NPN"/>NPN. Bla bla bla bla bla bla bla bla
        bla bla bla bla bla bla bla bla bla bla bla bla
        bla bla bla bla bla bla bla bla bla bla bla bla
        bla bla bla.</p>

    </div>
    <div>
        <head>Seconda divisione</head>

        <p>Bla bla bla bla bla bla bla bla bla bla bla bla
        bla bla bla bla bla bla bla bla bla bla bla bla
        bla bla bla bla bla bla bla bla bla bla bla bla
        bla.</p>
    </div>
    <divGen type="index" n="Index" />
```



## 53.4 Riferimenti


- *SGMLtools lite*, http://sgmltools-lite.sourceforge.net/
- Ian Jackson, Arno van Rangelrooij, *Debiandoc-SGML Markup Manual*
- *The DocBook DTD*, http://www.oasis-open.org/docbook/
- David Rugge, Mark Galassi, Eric Bischoff, *Writing documentation using DocBook*, http://opensource.bureau-cornavin.com/crash-course/
- Norman Walsh, Leonard Muellner, *DocBook: The Definitive Guide*, 2001, O'Reilly & Associates, Inc., ISBN 1-56592-580-7, http://oreilly.com/catalog/docbook/chapter/book/docbook.html
- TEI Consortium, *Text Encoding Initiative*, http://www.tei-c.org/index.xml
- Martin Mueller, *A very gentle introduction to the TEI*, http://wayback.archive.org/web/*/http://www.tei-c.org/Sample_Manuals/mueller-main.htm
- TEI Consortium, *TEI Lite DTD*, http://www.tei-c.org/Guidelines/Customization/Lite/DTD/teixlite.dtd
- Sebastian Rahtz, *PassiveTeX*, http://projects.oucs.ox.ac.uk/passivetex/index.xml , http://projects.oucs.ox.ac.uk/passivetex/passivetex.zip


[1] Per fare un esempio evidente, basta pensare all'inserzione di immagini e a ciò che si può ottenere in un formato finale puramente testuale: niente immagini.

[2] Non ha senso nella traduzione HTML.

[3] L'elemento '**htmlurl**' crea qualche problema quando si vogliono indicare caratteri speciali nell'URI, come nel caso della tilde. Sotto questo aspetto, per evitare problemi, è meglio limitarsi all'uso di '**url**'.

[4] LinuxDoc cerca di privilegiare in qualche modo l'ambiente matematico di LaTeX. Per richiamarlo è sufficiente delimitarlo attraverso le parentesi quadre, che così non possono essere usate in modo letterale. Come nel caso di altri simboli speciali, anche le parentesi quadre vanno indicate con l'uso di macro.

[5] Si osservi che in questo caso si tratterebbe di una didascalia vera e propria e non di un titolo come si fa nell'elemento '**figure**'.