```
$ nsgmls -s file.html
```

ISBN 978-88-905012-2-7

Via Morganella Est, 21 -- I-31050 Ponzano Veneto (TV) -- appunti2@gmail.com

L'opera, nei file sorgenti e nella composizione finale, include degli esempi in forma di sequenze animate, contenenti eventualmente anche delle spiegazioni vocali. Si tratta di video brevi e di qualità molto bassa. Tuttavia, a seconda di come viene diffusa o fruita l'opera, può darsi che sia necessario assolvere a degli obblighi di legge.

Il numero ISBN 978-88-905012-2-7 si riferisce all'opera originale in formato elettronico, pubblicata a titolo gratuito.

Se quest'opera viene consultata in-linea, attraverso uno spazio web, le informazioni generali sull'accesso ai file dell'opera (indirizzo IP di origine, nome del provider della connessione di origine, nome e versione del navigatore usato per accedere, geolocalizzazione dell'origine, data e orario di accesso), assieme al dettaglio delle pagine visitate o dei file scaricati, potrebbero essere annotate in un registro (log) di tale spazio web. Ciò per fini di controllo e statistica, a qualunque titolo. La conservazione di tale registro, se presente, dipende dalla politica del gestore e potrebbe avvenire per una durata di tempo indeterminata.

Quest'opera non contiene tecnologie atte a raccogliere e annotare i dati personali degli utenti che la consultano. Tuttavia, lo spazio web che la ospita potrebbe richiedere una forma di iscrizione o di autenticazione. Se tale iscrizione o autenticazione fosse richiesta, è necessario valutare l'informativa sul trattamento dei dati personali dello spazio web in questione, per sapere come e a che scopo tali dati vengono trattati.

Quest'opera, così come realizzata dal suo autore, non contiene inserzioni pubblicitarie. Tuttavia, lo spazio web che la ospita potrebbe iniettare il codice necessario a somministrare della pubblicità durante la sua consultazione o prima dello scarico dei file. Tali inserzioni pubblicitarie, se ci sono, non hanno nessuna relazione con l'autore di quest'opera e nemmeno vi portano alcun beneficio economico, in quanto servono esclusivamente al mantenimento dello spazio web ospitante.
Le inserzioni pubblicitarie, se presenti, possono utilizzare una tecnologia atta a riconoscere gli accessi che provengono dallo stesso computer o dallo stesso terminale, assieme a tutte le informazioni che possono essere estrapolate dall'origine dell'accesso e sulle funzionalità del computer o del terminale usato per accedere (incluso il fatto che sia disponibile o meno del software che possa essere utile a recepire la pubblicità stessa). Per conoscere il modo in cui le informazioni vengono raccolte dalla pubblicità (se c'è) e il loro utilizzo effettivo, è necessario valutare l'informativa sul trattamento dei dati personali dello spazio web che ospita l'opera.

2

Una copia della licenza GNU General Public License, versione 3, si trova nell'appendice A; una copia della licenza GNU Free Documentation License, versione 1.3, si trova nell'appendice B; una copia della licenza Creative Commons Attribution-ShareAlike, versione italiana 2.5, si trova nell'appendice C.

A copy of GNU General Public License, version 3, is available in appendix A; a copy of GNU Free Documentation License, version 1.3, is available in appendix B; a copy of Creative Commons Attribution-ShareAlike License, italian version 2.5, is available in appendix C.

Per tutti i riferimenti dell'opera si veda *http://informaticalibera.net* . Al momento della pubblicazione di questa edizione, i punti di distribuzione in-linea più importanti, sono presso Internet Archive (*http://www.archive.org/details/AppuntiDiInformaticaLibera*), il GARR (*http://appuntilinux.mirror.garr.it/mirrors/appuntilinux/*), ILS (*http://appunti.linux.it*) e il Pluto (*http://a2.pluto.it*).

3

Capitolo 47

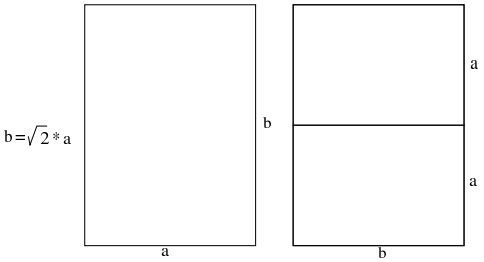
Editoria e stile

`.dictrc` 72 `ascii` 49 `aspell` 63 `buildhash` 61 `dict` 72 `dict.conf` 72 `dictd` 69 `gucharmap` 52 `hunspell` 63 `iconv` 55 `ispell` 58 `luit` 57 `munchlist` 61 `recode` 54 `unicode` 51

47.1 Formati standard della carta

Lo standard ISO 216, così come gli standard UNI 936 e DIN 476, definisce i formati di carta più comuni, secondo una logica molto semplice: i lati del foglio di carta hanno un rapporto fisso, dove il lato lungo è pari alla radice quadrata di due (circa 1,4142) per la lunghezza del lato corto (figura 47.1).

Figura 47.1. Il rapporto tra i lati di un foglio ISO 216.



Questo rapporto ha una proprietà importante, che consente al foglio di carta di essere dimezzato sul lato lungo, oppure di essere raddoppiato sul lato corto, mantenendo lo stesso rapporto tra i lati.

Lo standard ISO 216 definisce tre diverse serie di questi formati, ognuna delle quali parte da una dimensione di partenza, generando le altre dimensioni suddividendo quella precedente a metà, sul lato lungo. La serie A, ha come punto di riferimento il formato A0, corrispondente a un foglio con un'area di un metro quadro, tuttavia non si tratta del formato più grande, che è ottenuto raddoppiando due volte il formato A0, ottenendo così quattro metri quadri.

La tabella 47.2 elenca le dimensioni di tutti i formati delle tre serie, denominate A, B e C. Come si può osservare, i valori sono approssimati al millimetro, in aderenza al SI (sezione 47.3.3).

Tabella 47.2. ISO 216: formato A, B e C.

A	mm	B	mm	C	mm
4A0	1682 × 2378	--	--	--	--
2A0	1189 × 1682	--	--	--	--
A0	841 × 1189	B0	1000 × 1414	C0	917 × 1297
A1	594 × 841	B1	707 × 1000	C1	648 × 917
A2	420 × 594	B2	500 × 707	C2	458 × 648
A3	297 × 420	B3	353 × 500	C3	324 × 458
A4	210 × 297	B4	250 × 353	C4	229 × 324
A5	148 × 210	B5	176 × 250	C5	162 × 229
A6	105 × 148	B6	125 × 176	C6	114 × 162
A7	74 × 105	B7	88 × 125	C7	81 × 114
A8	52 × 74	B8	62 × 88	C8	57 × 81
A9	37 × 52	B9	44 × 62	C9	40 × 57
A10	26 × 37	B10	31 × 44	C10	28 × 40

47.1.1 Utilizzo pratico dei vari formati ISO 216

Tabella 47.3. Esempi di utilizzo pratico dei vari formati.

Formati	Utilizzo
A0, A1	Disegno tecnico; poster.
A2, A3	Disegno; diagrammi; tabelle di grandi dimensioni.
A4	Lettere; riviste; cataloghi; carta per stampanti comuni e per fotocopiatrici.
A5	Blocchi per appunti.
C4	Buste per il formato A4.
C5	Buste per il formato A4 piegato a metà.
C6	Buste per il formato A4 piegato due volte.
B4, A3	Giornali.

La percentuale di ingrandimento o di riduzione di un formato per ottenerne un altro, si determina facilmente, tenendo conto che si sta facendo riferimento all'ampiezza e all'altezza del foglio, non alla sua area. In pratica, riducendo un formato A4 al 50 %, si ottiene un formato A6, mentre per arrivare al formato A5 occorre usare una riduzione al 71 %. In altri termini, 71 %, ovvero 0,71, approssima la radice quadrata di 0,5. La tabella 47.4 riepiloga alcune trasformazioni tipiche, da un formato a un altro dello standard ISO 216.

Tabella 47.4. Esempi di ingrandimento e riduzione dei formati più comuni.

Trasformazione richiesta	rapporto	percentuale (approssimata)
da A***n*** a A***n***+1	$\sqrt{0{,}5}$	71 %
da B***n*** a A***n***	$\sqrt{\sqrt{0{,}5}}$	84 %
da A***n*** a B***n***	$\sqrt{\sqrt{2}}$	119 %
da B***n*** a A***n***−1	$\sqrt{\sqrt{2}}$	119 %
da A***n*** a A***n***−1	$\sqrt{2}$	141 %

La massa di un foglio di serie A, può essere determinata facilmente, sapendo che A0 ha una superficie di un metro quadro. In pratica, basta conoscere la densità superficiale della carta (la cosiddetta grammatura) che si esprime normalmente in grammi per metro quadro, dividendone opportunamente il valore: l'A***n*** ha una massa pari a 2^{-n} volte quella dell'A0. Per esempio, la massa di un foglio A4 è 2^{-4} volte quella di un A0; ovvero 1/16; se la grammatura è 80 g/m^2, la massa di un foglio A4 è 5 g.

Le dimensioni dei fogli delle tre serie ISO 216 possono essere determinate anche attraverso delle formule matematiche, come mostrato nella tabella 47.5. Si osservi che le misure che si ottengono sono espresse in metri.

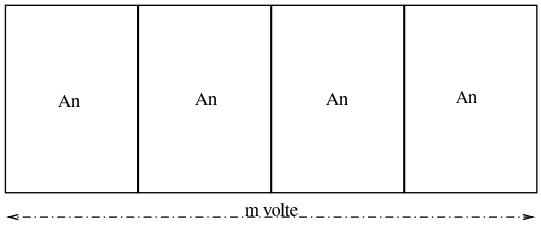
Tabella 47.5. Formule per calcolare le dimensioni della carta secondo lo standard ISO 216.

Formato	Ampiezza in metri	Altezza in metri
A***n***	$2^{(-1/4-n/2)}$	$2^{(1/4-n/2)}$
B***n***	$2^{(-n/2)}$	$2^{(1/2-n/2)}$
C***n***	$2^{(-1/8-n/2)}$	$2^{(3/8-n/2)}$

47.1.2 Formati multipli

Quando non si può utilizzare un formato in cui il rapporto tra la lunghezza dei lati sia quello delle serie A, B o C comuni, si possono usare dei multipli di uno di questi formati. Come si vede nella figura 47.6, si tratta di affiancare più fogli di un certo formato, estendendo il lato corto. Questi formati estesi si indicano come A***n***x***m***, dove ***m*** rappresenta quanti fogli di tipo A***n*** affiancare. Per esempio, il formato A3 è equivalente al formato A4x2.

Figura 47.6. Formati multipli A***n***x***m***.



47.2 Nozioni elementari di tipografia

Prima di studiare un programma di editoria elettronica conviene conoscere almeno qualche nozione di tipografia. Studiando la natura del problema si può comprendere la ragione di alcuni comportamenti dei programmi più raffinati che rispecchiano, nella loro impostazione, la filosofia della tipografia tradizionale.

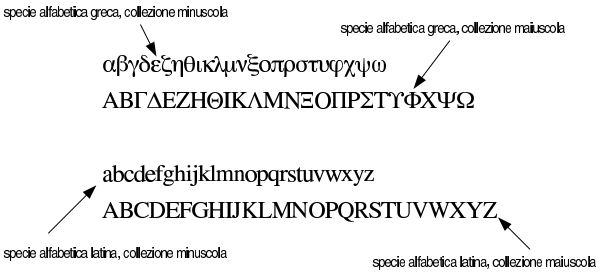
47.2.1 Carattere

Il ***carattere*** è qualunque segno grafico utilizzato in tipografia per rappresentare le lettere, i segni di interpunzione, le cifre e altri grafemi. La conoscenza delle caratteristiche fondamentali del carattere da stampa è utile per poter comprendere il funzionamento e la logica dei programmi di composizione tipografica. Sul carattere si possono distinguere diversi aspetti, in particolare: la specie alfabetica; lo stile, o il gruppo stilistico; la serie alfabetica, o la variante di serie; la scala dimensionale.

47.2.1.1 Specie alfabetica

La ***specie*** è una collezione di segni di un tipo di scrittura. Per quanto riguarda l'europa occidentale, la specie alfabetica comune è quella dell'alfabeto latino. All'interno di una specie alfabetica si possono distinguere diverse collezioni alfabetiche, per esempio come nella distinzione tra lettere maiuscole e minuscole che avviene nell'alfabeto latino.

Figura 47.7. Specie e collezione.



Dalla differenza tra gli alfabeti nasce a volte la necessità di rendere un testo attraverso un alfabeto alternativo. La ***traslitterazione*** è il procedimento di traslazione da un sistema alfabetico a un altro, in modo da ricomporre un testo facendo uso di un sistema alfabetico diverso da quello originale. La traslitterazione punta a riprodurre un testo in modo che sia possibile in qualsiasi momento il procedimento inverso per riottenere il testo originale. Il caso più comune in cui si ha la necessità di utilizzare la traslitterazione è quello della citazione in cui l'originale utilizza un alfabeto esotico per il quale non si dispone del carattere tipografico. Come si può immaginare, la traslitterazione è regolata da norme internazionali.

47.2.1.2 Gruppo stilistico

Una volta definita la specie di un carattere si possono distinguere delle varianti che riguardano lo ***stile***, ovvero il disegno e il suo gusto estetico. Sull'alfabeto latino sono stati realizzati una quantità così grande di stili diversi che è difficile persino riuscire a classificarli. In generale vi si fa riferimento attraverso il nome. Gli stili tradizionali comuni nella composizione elettronica sono: Times, Helvetica e Courier.

Figura 47.8. Stile.

I tre nomi citati rappresentano oggi, simbolicamente, le caratteristiche fondamentali di uno stile: la presenza o l'assenza di grazie e la proporzionalità o meno della larghezza dei segni.[1]

Le grazie sono dei piedini terminali che hanno lo scopo di abbellire il carattere e di guidare la vista durante la lettura. Il Times è il tipico stile con grazie, mentre Helvetica è il suo opposto.

I segni dei caratteri da stampa sono generalmente di larghezza diversa; solo le prime forme di scrittura meccanica, come la macchina da scrivere e le prime stampanti, hanno creato la necessità di utilizzare dei simboli a larghezza uniforme. Il Courier è il rappresentante di questo tipo di stile a larghezza fissa.

In generale, uno stile riguarda esclusivamente una specie alfabetica, ma quando uno stile assume importanza e notorietà, può succedere che venga adottato anche da altre specie. Per questo si può distinguere tra Times Roman (Times New Roman), Times Greco, Times Cirillico e altri. Il primo tra quelli citati è ovviamente il Times dell'alfabeto latino.

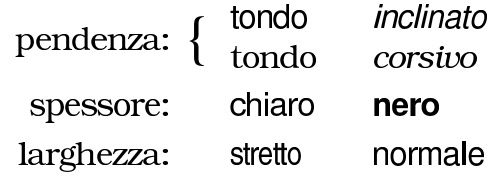
47.2.1.3 Serie

La ***serie alfabetica***, o la variante di serie, rappresenta una distinzione all'interno di uno stile, in base alla ***forma***. Le forme comuni di uno stesso stile riguardano la pendenza, il tono e la larghezza.

- La pendenza si riferisce all'inclinazione delle aste e si distingue generalmente tra ***tondo***, rappresentato da un carattere con aste verticali, e ***corsivo***, in cui le aste sono inclinate in avanti. Generalmente, l'aspetto dei caratteri di un corsivo, pur restando all'interno della stesso stile, è abbastanza diverso da quello del tondo. Quando si utilizza un sistema di composizione elettronico può capitare di avere a disposizione uno stile nel quale manchi il corsivo, ottenendolo però in qualche modo distorcendo il tondo. In questo caso si parla preferibilmente di carattere «inclinato» in modo volutamente generico.
- Il tono, o lo spessore, rappresenta l'intensità del carattere che si percepisce visivamente. Essendo un concetto che deriva dalla stampa con inchiostro nero, si distingue generalmente tra ***chiarissimo***, ***chiaro***, ***nero*** (***neretto***) e ***nerissimo***.
- La larghezza è una caratteristica di cui dispongono solo alcuni stili, ovvero li può riguardare direttamente, nel senso che uno stile per sua natura può essere «stretto» o «largo». In base alla larghezza si distinguono solitamente: lo ***strettissimo***, lo ***stretto***, il normale, il ***largo*** e il ***larghissimo***.

È bene chiarire che ogni stile può disporre o meno di varianti seriali adatte. Alcuni stili, spesso riferiti a specie alfabetiche simboliche, dispongono di una serie unica.[2]

Figura 47.9. Serie.



47.2.2 Tipometria

La tipometria è la misurazione degli elementi che riguardano la composizione e l'impaginazione. Le voci più importanti sono costituite dai corpi (l'altezza dei caratteri), dalla spaziatura, dall'interlinea, dalla giustezza e dalla giustificazione. In breve, il corpo è l'altezza del carattere, la spaziatura è la distanza tra una parola e l'altra in una riga, l'interlinea è lo spazio verticale aggiuntivo tra le righe, la giustezza è lo spazio orizzontale che le righe di testo hanno a disposizione, la giustificazione è il procedimento di regolazione della spaziatura e dell'interlinea in modo da ottenere un allineamento delle righe con i margini (sia in orizzontale, sia in verticale).

47.2.2.1 Corpo, dimensioni e scala

La dimensione del carattere si misura in senso verticale e si definisce ***corpo***. Per misurare il corpo e le altre dimensioni che riguardano i caratteri si possono utilizzare diverse unità di misura, ma quando si tratta di sistemi di composizione elettronica a mezzo di software, è molto probabile che si disponga solo del pica e del punto anglo-americano:

- 1 pica = 1/6 di pollice;
- 1 punto = 1/12 di pica = 1/72 di pollice.

Per comprendere cosa sia il corpo di un carattere è bene descrivere le varie componenti dell'altezza di questo. La figura 47.10 mostra schematicamente la parola «Agglomerato» suddivisa secondo le componenti verticali della dimensione del carattere.

Figura 47.10. Le dimensioni del carattere.

Il carattere si appoggia su una linea che rappresenta la base della «parte mediana»; le lettere come la «l» si alzano occupando anche la «parte ascendente»; altre, come la «g», si allungano in basso a occupare la «parte discendente». Il corpo del carattere include anche uno spazio aggiuntivo: la «spalla». Si distingue una spalla superiore, costituita da uno spazio minimo sopra la parte ascendente, e la spalla inferiore che si trova al di sotto della parte discendente (nella figura la spalla è molto grande, in proporzione, rispetto alla realtà).

La distanza tra la base di una riga (la base della parte mediana) e la base di quella successiva dovrebbe essere superiore o al minimo uguale alla grandezza del corpo. Quando questa distanza è superiore, lo spazio aggiuntivo è l'***interlinea***. Con i sistemi di composizione elettronica per mezzo di software, si misura generalmente lo spazio tra le basi delle righe ed è ammissibile anche l'utilizzo di distanze inferiori all'altezza del carattere, ottenendo in pratica una sovrapposizione della parte mediana inferiore di una riga con la parte mediana superiore di quella successiva.

La rappresentazione di un carattere con un corpo di una data dimensione dipende dalla disponibilità di questo. Con i sistemi tipografici meccanici è necessario disporre di una serie di caratteri mobili differenti, distinti in base a una scala. Con i sistemi di composizione elettronica via software si possono trovare dei caratteri riproducibili in qualsiasi corpo, eventualmente generando dei file opportuni per la scala richiesta. Tuttavia, in presenza di dimensioni particolarmente piccole si rischia di perdere dei dettagli importanti dei segni che compongono lo stile utilizzato e, di conseguenza, potrebbe essere preferibile l'utilizzo di una variante dello stile che sia più adatta alle dimensioni ridotte.

47.2.2.2 Giustezza, spaziatura e giustificazione orizzontale

La giustezza è lo spazio orizzontale a disposizione delle righe di testo; in altri termini, è la larghezza della colonna all'interno della quale si può distribuire il testo. La spaziatura è lo spazio tra la fine di una parola e l'inizio di quella successiva.

Nei testi in italiano, la spaziatura è uniforme, senza eccezioni, a differenza della tradizione tipografica di altri paesi. Per esempio, la spaziatura dopo un punto fermo è esattamente uguale a quella di qualunque altra situazione. Quando si utilizza il sistema di composizione TeX per scrivere un testo in italiano, si dovrebbe inserire il comando '`\frenchspacing`' per evitare anomalie nella spaziatura.

Quando si vuole ottenere un allineamento del testo all'inizio e alla fine della giustezza, si parla di giustificazione (orizzontale). Per ottenerla, è necessario che la spaziatura sia adattata in modo da arrivare a questo risultato. La giustificazione orizzontale è solo una delle scelte stilistiche che il tipografo ha a disposizione: non si tratta di una convenzione obbligatoria.

47.2.2.3 Giustificazione verticale

Come nel caso della giustificazione orizzontale, ci può essere la necessità o l'opportunità di adattare l'interlinea in modo da riempire completamente le pagine. Ciò si ottiene attraverso la giustificazione verticale.

47.2.3 Il carattere nel software di composizione

Utilizzando i programmi di composizione tipografica si è costretti generalmente a fare i conti con la terminologia dei paesi di lingua inglese e con altri problemi legati alla rappresentazione simbolica dei segni all'interno del software. La tradizione tipografica di questi ha generato dei termini che non sono perfettamente traducibili con concetti della tradizione italiana, per cui si utilizzano alcuni termini di origine anglofona, eventualmente tradotti in modo letterale.

47.2.3.1 Terminologia

In inglese si utilizza normalmente il termine *font* per fare riferimento al carattere tipografico. Spesso non si traduce questo termine in qualcosa che riguardi la tradizione tipografica italiana, mantenendo piuttosto il termine inglese invariato; tuttavia, alle volte viene utilizzata la forma: ***fonte***.

Se il contesto non richiede un'aderenza perfetta con il termine originale inglese, si possono usare forme espressive più comprensibili, come «carattere», «tipo di carattere», «carattere tipografico» o «carattere da stampa».

47.2.3.2 Caratteristiche di una carattere tipografico elettronico

Il carattere tipografico usato nel software applicativo di composizione, ha una serie di caratteristiche, alcune delle quali sono fondamentali.

- ***foundry***, ***fonderia***

  La fonderia è il produttore del carattere tipografico, cioè chi ha creato la tipizzazione,[3] pur senza esserne il disegnatore. Per fare un esempio comune, Adobe è la fonderia dello stile Times New Roman.

- ***family***, ***famiglia***

  La famiglia del carattere, inteso come traduzione del termine *font family*, corrisponde simultaneamente alla specie e allo stile del carattere. In altri termini, rappresenta sia la specie alfabetica, sia lo stile. Per fare un esempio, la famiglia Times New Roman è un carattere di specie latina e di stile Times.

- All'interno di una famiglia si distinguono normalmente le serie riferite alla forma: spessore (*weight*), inclinazione (*slant*) e larghezza (*set*, o *width*).

- ***codifica***

  La codifica rappresenta l'elemento nuovo più importante nelle caratteristiche di un carattere tipografico per l'elaborazione via software. Il problema viene descritto nella prossima sezione.

47.2.3.3 Codifica

L'utilizzo dei caratteri con i sistemi di composizione basati sul software richiede un abbinamento tra segni e simboli binari. Questo

abbinamento è definito dalla codifica. Il problema si può intendere meglio se si pensa a un programma a composizione differita.

In questi casi si parte da un file sorgente, scritto probabilmente secondo la codifica UTF-8, con il quale il programma deve comporre il risultato, utilizzando i caratteri a disposizione.

Il carattere tipografico utilizzato dal programma di composizione è contenuto normalmente all'interno di file, da cui questo programma estrae le informazioni necessarie attraverso un riferimento dato da un codice numerico. In condizioni normali, il programma di composizione fa riferimento al simbolo binario utilizzato nel sorgente per ottenere il segno corrispondente all'interno del carattere tipografico utilizzato (eventualmente attraverso una qualche traslazione). In pratica, alla lettera «A» nel sorgente dovrebbe corrispondere la lettera «A» del carattere tipografico che si sta utilizzando, ma se il carattere tipografico è organizzata in modo differente, si potrebbe ottenere qualcosa di diverso. Questo problema si avverte di solito quando si utilizza una famiglia di caratteri che fa riferimento a una specie simbolica, o comunque a un alfabeto che non ha alcuna corrispondenza con la codifica utilizzata nel sorgente. In questi casi, di solito, per rappresentare i segni si può fare uso di comandi speciali interpretati opportunamente dal programma di composizione.

Un programma di composizione potrebbe disporre di caratteri tipografici che hanno solo una corrispondenza parziale con la codifica utilizzata per scrivere il sorgente, per esempio, potrebbero mancare alcuni segni che vengono messi a disposizione attraverso altre specie.

Il problema viene riproposto nella sezione 47.6.

47.2.4 Problemi legati ai caratteri

Nelle origini della tipografia, molti caratteri mobili rappresentavano l'unione di più lettere o altri segni in logotipo (cioè l'unione in un simbolo unico). L'unione di questi derivava da delle consuetudini stilistiche o dalla forma dei segni adiacenti che per qualche motivo potevano richiedere un avvicinamento o un adattamento.

Il ***legato*** (in inglese *ligature*) è l'unione di due o più segni per motivi storici o estetici; i più comuni sono le sequenze «fi», «fl» e «ffi», dove le lettere vengono avvicinate in modo particolare fino a unirsi o a inglobarsi. Alcune forme di legato si sono tradotte in segni indipendenti, come nel caso di «AE» che si è trasformato in «Æ», «sz» che nella lingua tedesca è ormai «ß», «et» (latino) che è divenuto «&», ovvero l'attuale e-commerciale, e anche «ad» (latino), che nella lingua inglese è diventato «@» (*at*).

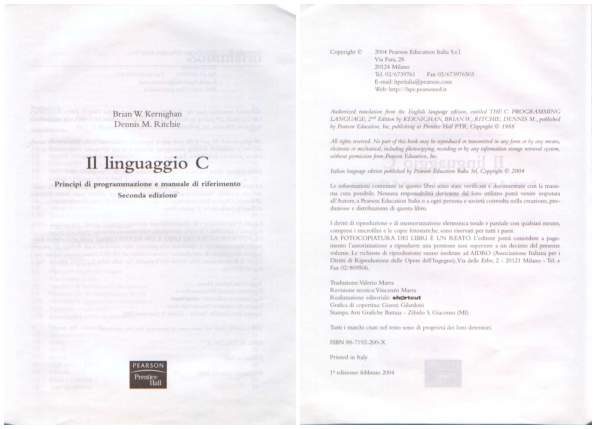
L'avvicinamento delle lettere, è motivato dalla forma di queste, per evitare il formarsi di vuoti visivi che potrebbero creare difficoltà alla lettura. I casi più comuni sono le sequenze «AV», «AT», «AY».

47.2.5 Il libro

Alcune componenti del libro, nella sua forma cartacea, hanno dei nomi particolari, che spesso si ignorano. Le figure successive descrivono sommariamente tali componenti, utilizzando come esempio il libro *Il linguaggio C* di Brian W. Kernigham e Dennis M. Ritchie, edizione Pearson, 2004.

Figura 47.11. Ciò che appare esternamente a un libro è: la ***prima di copertina*** (immagine a sinistra), o semplicemente «copertina», costituente la prima pagina assoluta; la ***quarta di copertina*** (immagine a destra) è l'ultima pagina assoluta; il ***dorso***, o ***costola*** (immagine in basso), è ciò che si trova in corrispondenza della rilegatura dei fogli, con le informazioni essenziali del libro, utili per identificarlo quando viene riposto in uno scaffale.

Figura 47.12. Immagine a sinistra: il ***frontespizio*** è una delle prime pagine del libro, la quale riprende le informazioni della copertina. Immagine a destra: la pagina che contiene le informazioni sulla pubblicazione (i diritti dell'opera, le sue varie edizioni, ecc.) è il ***colofone*** o *colophon* (la dizione *colophon* viene dal latino, ma il termine originario è *kolophon* e viene dal greco). Può trovarsi alla fine dell'opera oppure tra le primissime pagine.



Oltre agli esempi che appaiono nelle figure, vanno considerati anche la ***seconda di copertina***, che è la parte interna della prima di copertina, così come la ***terza di copertina***, che è la parte interna della quarta di copertina. Nei libri comuni, la seconda e la terza di copertina sono bianche, o comunque prive di contenuti, mentre nelle riviste diventano posizioni significative per la collocazione della pubblicità.

Lo specchietto successivo consente di abbinare alcuni dei termini descritti alla forma inglese corrispondente:

prima di copertina	*front cover page*
seconda di copertina	*inside front cover page*
terza di copertina	*inside back cover page*
quarta di copertina	*back cover page*

47.3 Stile letterario[4]

Il concetto di ***stile letterario*** potrebbe essere espresso semplicemente spiegando l'esigenza di realizzare un documento ***uniforme***: sia dal punto di vista visivo, sia dal punto di vista espressivo. Questo

coinvolge quindi l'aspetto grammaticale (ortografia, sintassi, lessico, ecc.) e l'aspetto tipografico (impaginazione, tipi di carattere, dimensione, ecc.) o artistico.

L'esigenza di un'uniformità visiva deriva dal piacere e dal rilassamento che può dare al lettore un documento impaginato e strutturato in un modo ordinato e chiaro, per la facilità nella lettura che ne deriva. Nello stesso modo è importante l'uniformità grammaticale, cosa particolarmente delicata in una lingua come quella italiana in cui sono consentite molte variazioni, data la ricchezza linguistico-culturale delle varie regioni.

Il problema dell'uniformità stilistica si accentua quando si deve collaborare alla realizzazione di un progetto letterario. L'uniformità non è più solo un fatto di coerenza personale, ma di coerenza complessiva di tutto il gruppo.

47.3.1 Regole di composizione del testo

Il modo migliore per definire uno stile grammaticale è lo studio su un testo di grammatica. Nelle sezioni successive si raccolgono solo alcuni punti essenziali che non possono essere ignorati.

47.3.1.1 Punteggiatura e spaziatura

La punteggiatura si compone di quei simboli che consentono di separare le parole e di delimitare le frasi.

- Ogni parola è separata da un solo spazio.

  Tipograficamente, lo spazio è una separazione di ampiezza non definita, spesso ampliato o compresso, per ottenere un allineamento del testo, sia a sinistra, sia a destra. Un autore non deve pensare a queste cose quando scrive la propria opera; si deve limitare a spaziare le parole con un solo carattere spazio.[5]

  La dattilografia insegnava a ottenere testi allineati a sinistra e a destra con l'inserzione opportuna di spazi aggiuntivi, vicino alle parole composte da poche lettere (congiunzioni, articoli, ecc.). Questo tipo di tecnica è ormai da abbandonare, lasciando semmai che siano i programmi di composizione a prendersi cura di tali problemi, anche quando il risultato finale deve essere un file di testo puro e semplice.

  I programmi di composizione più evoluti facilitano il compito dello scrittore eliminando gli spazi superflui, per cui con questi non c'è l'esigenza di porre attenzione alla dimensione delle spaziature.[6]

- Il simbolo di punteggiatura normale è attaccato alla parola che precede ed è separato con uno spazio dalla parola che segue.

  Si tratta di: punto, virgola, due punti, punto e virgola, punto interrogativo e punto esclamativo.

  Alle volte, l'autore di documenti tecnici di informatica si lascia confondere dall'uso che si fa di tali simboli in un particolare linguaggio di programmazione o in altri ambiti analoghi. È chiaro, per esempio, che se si deve indicare un'estensione di un file, come «.sgml», non si può rispettare tale regola, in quanto il punto che precede quell'estensione non rappresenta un simbolo di punteggiatura del testo.

- Le parentesi sono attaccate al testo che racchiudono e, rispetto alla punteggiatura esterna, si comportano come un'unica parola.

  La parentesi di apertura è separata con uno spazio dalla parola che precede, mentre quella di chiusura è separata con uno spazio dalla parola che segue. I simboli di punteggiatura normale che dovessero seguire una parentesi chiusa vanno attaccati a questa ultima.

  Nella lingua italiana non è consentito racchiudere all'interno di parentesi un periodo terminante con il punto fermo (al contrario di quanto avvenga con la lingua inglese).

- Il testo riportato tra virgolette si comporta come quello racchiuso tra parentesi.

  La lingua italiana prevede l'uso di virgolette uncinate (in basso), virgolette elevate doppie e singole. Secondo la grammatica, le virgolette uncinate, o virgolette basse, sono da preferire. Tuttavia, dal momento che le virgolette elevate possono essere ottenute anche utilizzando soltanto il codice ASCII tradizionale a 7 bit, molti autori preferiscono accontentarsi e utilizzare solo quelle elevate.[7]

- Il trattino di unione è corto e unito alle parole da collegare.

  Si usa per unire insieme due parole in modo da formare una parola composta. I programmi di composizione tendono a considerare un trattino singolo come un trattino corto, proprio per questo scopo.

- La lineetta, o trattino lungo, serve per introdurre un discorso diretto, oppure un inciso.

  Il trattino utilizzato per delimitare un discorso diretto, viene usato normalmente solo in apertura. Può apparire anche un trattino in chiusura quando al discorso diretto segue un commento. Se il trattino si usa per delimitare un inciso, si usa per aprirlo e solitamente anche per chiuderlo, come se si trattasse di parentesi.

  Generalmente, il trattino lungo è preceduto e seguito da uno spazio; davanti al trattino di chiusura vanno collocati il punto interrogativo, il punto esclamativo e i puntini, mentre per gli altri simboli di punteggiatura non esiste una convenzione precisa.[8]

47.3.1.2 Utilizzo dei simboli di interpunzione

L'uso della punteggiatura nella lingua italiana è definito da regole molto vaghe che si prestano a facili eccezioni di ogni tipo. Qui si elencano solo alcuni concetti fondamentali.

,	La virgola è un segno di interpunzione che collega due segmenti di testo separati da una pausa debole.
;	Il punto e virgola è un segno di interpunzione che si colloca a metà strada tra la virgola e il punto. Non segna la chiusura di un periodo.
:	I due punti sono un simbolo di interpunzione ***esplicativo***. Collegano due segmenti di testo separati dal punto di vista sintattico, in cui la seconda parte, quella che segue il simbolo, elenca, chiarisce o dimostra il concetto espresso nella prima parte.
.	Il punto fermo è un segno di interpunzione che collega due segmenti di testo separati da un pausa forte. Generalmente segna la conclusione di un periodo. La parola successiva al punto ha l'iniziale maiuscola.
!	Il punto esclamativo indica generalmente la conclusione di un'esclamazione affermativa. Generalmente, quando conclude un periodo, il testo che segue ha l'iniziale maiuscola.
?	Il punto di domanda indica un tono interrogativo alla fine di una frase. Generalmente, quando conclude un periodo, il testo che segue ha l'iniziale maiuscola.
...	I punti di sospensione sono in numero fisso di tre e indicano che il discorso non viene portato a conclusione. Generalmente, sono uniti alla parola o al segno di interpunzione che li precede, oppure distanziati, a seconda che siano solo una sospensione oppure indichino l'omissione di un nome o di un'altra parola. Se si trovano alla fine di un periodo, dove andrebbe collocato un punto, questo non viene aggiunto e la frase successiva inizia con la maiuscola. Nello stesso modo, se si trovano alla fine di un'abbreviazione che termina con un punto, questo punto viene assorbito.
ecc**.**	Il punto di abbreviazione, quando si trova alla fine di un periodo, conclude da solo anche il periodo stesso, ed è seguito da iniziale maiuscola.
( )	Le parentesi, generalmente tonde, servono per delimitare un inciso, come un commento, una nota dello scrivente, un chiarimento, ecc. Generalmente, i commenti del redattore o del traduttore sono terminati, entro l'ambito delle parentesi, con le sigle NdR (nota del redattore) e NdT (nota del traduttore).

47.3.1.3 Accenti e troncamenti

Nella lingua italiana scritta, l'uso degli accenti è un fatto puramente convenzionale. Ciò significa che l'accento non indica necessariamente il suono che ha effettivamente la lettera accentata, ma solo la sua rappresentazione consueta (più avanti, nella sezione 47.3.1.4 è riportato il testo originale della norma UNI 6015 sul «segnaccento obbligatorio»).[9]

- Nella lingua scritta è prevista (ed è obbligatoria) solo l'accentazione delle vocali finali delle parole nelle quali il tono della voce si rafforza sull'ultima sillaba (accento grafico).
  È possibile l'uso dell'accento per le vocali interne quando ciò serva per togliere ambiguità tra termini omografi (scritti nello stesso modo) che abbiano significati differenti. Generalmente, questa ambiguità è risolta dal contesto e raramente si incontra la necessità di utilizzare accenti interni.
- Si utilizza comunemente solo l'accento grave (**àèìòù**), con l'eccezione della vocale «e» che può avere l'accento acuto (**é**).
- Vogliono l'accento acuto le parole terminanti in **ché** (perché, poiché, ecc.), oltre a **né** (congiunzione) e **sé** (pronome tonico). In particolare, **sé** viene scritto generalmente senza accento quando è seguito da **stesso**, anche se la grammatica non lo richiede.
- Vogliono l'accento alcuni monosillabi contenenti due vocali: **ciò**, **già**, **giù**, **più** e **può**.
- Vogliono l'accento i monosillabi che senza potrebbero avere un significato differente. La tabella 47.15 mostra l'elenco dei monosillabi accentati più importanti.
- Non vogliono l'accento alcuni monosillabi tra cui: **qui**, **qua**, **sto** e **sta**.
- Solo alcune parole tronche richiedono la segnalazione di tale troncamento con l'apostrofo finale. In particolare: **po'** (poco), **mo'** (modo), **ca'** (casa) e alcuni imperativi.
- L'accento circonflesso (^) non si usa più. Serviva per i nomi terminanti in **−io** che al plurale terminerebbero in **−ii** (per esempio: armadio, armadii). Attualmente, si tende a usare questi plurali con una sola **−i** finale, a parte i casi in cui ciò genera ambiguità (**assassino**, **assassini**; **assassinio**, **assassinii**).

Tabella 47.15. Elenco dei monosillabi accentati più importanti e dei loro equivalenti (omografi) non accentati.

dà	indicativo di dare (dà valore)	da	preposizione (da voi)
è	verbo	e	congiunzione
là	avverbio (resta là)	la	articolo
lì	avverbio (vado lì)	li	pronome
né	congiunzione (né questo né quello)	ne	pronome (ne voglio ancora)
sé	pronome tonico (pieno di sé)	se	pronome atono o congiunzione
sì	avverbio (dice di sì)	si	pronome

Alle volte, l'uso delle vocali accentate può creare problemi tecnici, dovuti alla loro mancanza nell'insieme di caratteri a disposizione. Nelle circostanze in cui non è possibile scrivere direttamente con lettere accentate (per esempio quando si dispone di un sistema configurato male, o la tastiera non dispone dei simboli necessari), occorre utilizzare delle tecniche di rappresentazione che dipendono dal programma utilizzato per la composizione.

Con l'introduzione dell'insieme di caratteri universale, che nei sistemi Unix si attua attraverso la codifica UTF-8, il problema della rappresentazione dei caratteri con lettere accentate, viene meno, qualunque sia la lingua da utilizzare.

Riquadro 47.16. Vocali accentate attraverso l'uso di macro SGML e XML.

Vocale accentata	Macro corrispondente	Vocale accentata	Macro corrispondente
à	à	À	À
è	è	È	È
ì	ì	Ì	Ì
ò	ò	Ò	Ò
ù	ù	Ù	Ù
é	é	É	É

Riquadro 47.17. Vocali accentate per TeX.

Vocale accentata	Codice TeX corrispondente	Vocale accentata	Codice TeX corrispondente
à	\`a	À	\`A
è	\`e	È	\`E
ì	\`{\i}	Ì	\`I
ò	\`o	Ò	\`O
ù	\`u	Ù	\`U
é	\'e	É	\'E

Riquadro 47.18. Trucco per rappresentare le vocali accentate quando non si può fare altrimenti.

Vocale accentata	Vocale apostrofata corrispondente	Vocale accentata	Vocale apostrofata corrispondente
à	a‘	À	A‘
è	e‘	È	E‘
ì	i‘	Ì	I‘
ò	o‘	Ò	O‘
ù	u‘	Ù	U‘
é	e’	É	E’

47.3.1.4 Segnaccento obbligatorio (UNI 6015)

Quello che segue è la norma UNI 6015 sull'uso degli accenti. Il testo è stato ottenuto da *Grafica; scienza, tecnologia e arte della stampa e della comunicazione*, *Preparazione del manoscritto* http://www.apenet.it/ .

Segnaccento obbligatorio nell'ortografia della lingua italiana (Uni 601567):

1. *Scopo*

La presente unificazione ha lo scopo di stabilire le regole ortografiche per il segnaccento nei testi stampati in lingua italiana, quando esso sia obbligatorio.

2. *Definizione*

2.1 Il segnaccento (o segno d'accento, o accento scritto) serve a indicare esplicitamente la vocale tonica, per esempio: *andrà*, *colpì*, *temé*, *virtù*.

2.2. Il segnaccento può essere grave (‘’) o acuto (‘’).

3. *Uso*

Il segnaccento è obbligatorio nei casi seguenti:

3.1. Su alcuni monosillabi, per distinguerli da altri monosillabi che si scrivono con le stesse lettere ma senza accento:

ché («poiché», congiunzione causale) per distinguerlo da *che* (congiunzione in ogni altro senso, o pronome);

dà (indicativo presente di dare) per distinguerlo da *da* (preposizione) e *da'* (imperativo di dare);

dì («giorno») per distinguerlo da *di* (preposizione) e *di'* (imperativo di dire);

è (verbo) per distinguerlo da *e* (congiunzione);

là (avverbio) per distinguerlo da *la* (articolo, pronome, nota musicale);

lì (avverbio) per distinguerlo da *li* (articolo, pronome);

né (congiunzione) per distinguerlo da *ne* (pronome, avverbio);

sé (pronome tonico) per distinguerlo da *se* (congiunzione, pronome atono);

sì («così», o affermazione) per distinguerlo da *si* (pronome, nota musicale);

té (pianta, bevanda) per distinguerlo da *te* (pronome).

3.2. Sui monosillabi: *chiù*, *ciò*, *diè*, *fé*, *già*, *giù*, *piè*, *più*, *può*, *scià*.

3.3. Su tutte le parole polisillabe su cui la posa della voce cade sulla vocale che è alla fine della parola, per esempio: *pietà*, *lunedì*, *farò*, *autogrù*.

4. *Forma*

4.1. Il segnaccento, nei casi in cui è obbligatorio, è sempre grave sulle vocali: *a*, *i*, *o*, *u*.

4.2. Sulla *e*, il segnaccento obbligatorio è grave se la vocale è aperta, è acuto se la vocale è chiusa:

- è sempre grave sulle parole seguenti:

ahimè e *ohimè*, *caffè*, *canapè*, *cioè*, *coccodè*, *diè* e *gilè*, *lacchè*, *piè*, *tè*; inoltre sulla maggior parte dei francesismi adattati, come *bebè*, *cabarè*, *purè*, ecc. e sulla maggior parte dei nomi propri, come *Giosuè*, *Mosè*, *Noè*, *Salomè*, *Tigrè;*

- è acuto sulle parole seguenti:

ché («poiché») e i composti di *che* (*affinché*, *macché*, *perché*, ecc.), *fé* e i composti *affé*, *autodafé*, i composti di *re* e di *tre* (*viceré*, *ventitré*), i passati remoti (*credé*, *temé*, ecc., escluso *diè*), le parole *mercé*, *né*, *scimpanzé*, *sé*, *testé*.

4.3. Anche per la *o* si possono distinguere i due timbri (aperto o chiuso) con i due accenti (grave ed acuto) ma solo in casi in cui l'accento è facoltativo, per esempio: *còlto* (participio passato di *cogliere*), e *cólto* («istruito»).

47.3.1.5 Uso della «d» eufonica

Le congiunzioni **e**, **o** e la preposizione **a**, consentono l'aggiunta di una **d** eufonica, per facilitarne la pronuncia quando la parola che segue inizia per vocale. Si tratta di una possibilità e non di una regola; di questa **d** si potrebbe benissimo fare a meno.

Ognuno tende a usare questa **d** eufonica in modo differente, a seconda della propria cadenza personale, che ne può richiedere o meno la presenza. Quando si scrive, bisognerebbe mantenere lo stesso stile, anche sotto questo aspetto, quindi ognuno deve stabilire e seguire un proprio modo.

Esiste tuttavia un suggerimento che punta all'uso moderato di queste **d** eufoniche: usare la **d** solo quando la vocale iniziale della parola successiva è la stessa; e non usarla nemmeno quando, pur essendoci la stessa vocale iniziale nella parola successiva, ci sia subito dopo una **d** che possa complicare la pronuncia.

47.3.1.6 Elisione davanti alla lettera «h»

In linea di massima, l'articolo che si mette davanti a un termine che inizia con la lettera **h**, è quello che si userebbe pronunciando quella parola come se iniziasse per vocale. Secondo questo principio, va usata l'elisione, così come si fa con i termini che iniziano per vocale, senza alcuna «h» anteriore. Per esempio: l'harem; l'hotel; l'host.

Tuttavia, quando si tratta di un termine che, proveniendo da un'altra lingua, non è ancora diventato di uso comune e nella lingua originale si pronuncia con la lettera «h» iniziale aspirata, si preferisce evitare l'elisione.

47.3.1.7 Uso delle maiuscole

L'iniziale maiuscola si utilizza all'inizio del periodo e per evidenziare i nomi propri. La lingua italiana fa un uso diverso delle maiuscole rispetto ad altre lingue, limitandole al minimo indispensabile; pertanto, nelle situazioni di dubbio, è meglio utilizzare le lettere minuscole.

47.3.1.8 Plurali

Ci sono alcuni aspetti del plurale nella lingua italiana che vale la pena di annotare. In particolare, nel caso di chi deve utilizzare anche termini stranieri, si pone il problema di decidere se questi siano invariabili o meno. A questo proposito, esistono due regolette semplici e pratiche:

- le parole terminanti per consonante sono invariate al plurale;
- i termini di provenienza straniera non ancora assimilati sono invariati al plurale.

In particolare, per quanto riguarda la seconda, la logica è che non si può applicare un plurale secondo le regole di una lingua straniera mentre si usa l'italiano. Inoltre, dato che nella maggior parte dei casi si tratta di termini inglesi, che nella loro lingua prenderebbero quasi sempre una terminazione in **−s** al plurale, diventerebbe anche difficile la loro pronuncia in italiano.

47.3.1.9 Interfacce o interfaccie?

Esiste una regoletta che permette di stabilire facilmente come debba essere ottenuto il plurale delle parole che terminano in **−cia** e **−gia**: la **i** rimane se la **c** e la **g** sono precedute da vocale, oppure se la **i** viene pronunciata con accento, mentre viene eliminata se queste consonanti sono precedute da un'altra consonante.

Quindi si ha: **camicia**, **camicie** e **interfaccia**, **interfacce**; **ciliegia**, **ciliegie** e **spiaggia**, **spiagge**; **energia**, **energie**.

47.3.1.10 Elenchi

Gli elementi puntati, o numerati, possono essere composti da elementi brevi, oppure da interi periodi. Se tutti gli elementi sono brevi:

- l'elenco deve essere introdotto da una frase terminante con due punti;
- ogni elemento deve essere terminato con un punto e virgola, a eccezione dell'ultimo che termina normalmente con un punto.

La descrizione appena fatta mostra un esempio di elenco del genere. Se anche uno solo degli elementi è troppo lungo, è bene trasformare tutti gli elementi in periodi terminati da un punto. In tal caso, se l'elenco viene introdotto da una frase, anch'essa termina con un punto.

Ci possono essere situazioni in cui queste indicazioni non sono applicabili: come sempre è necessario affidarsi al buon senso.

47.3.1.11 Citazioni

Le citazioni, cioè le frasi o i brani riprodotti letteralmente da altri documenti, devono apparire distinte chiaramente dal testo normale. Si usano normalmente queste convenzioni:

- quando la citazione è incorporata nel testo viene delimitata attraverso le virgolette, oppure utilizzando il corsivo se la citazione è particolarmente breve;
- le citazioni incluse in un'altra citazione già virgolettata si evidenziano attraverso l'uso di un altro tipo di virgolette, cominciando da quelle uncinate («»), utilizzando poi quelle elevate doppie (‘‘’’) e terminando con quelle singole (‘’);
- quando la citazione è molto lunga e occupa diversi capoversi, conviene utilizzare un corpo minore o un altro espediente tipografico per distinguerla dal testo normale, come con l'uso di rientri differenti;
- quando la citazione è lunga e non si vogliono utilizzare altri espedienti per evidenziarla, si utilizzano le virgolette, ripetendo quelle di apertura all'inizio di ogni capoverso;

- all'interno delle citazioni possono apparire dei commenti o chiarimenti inseriti da chi scrive, delimitandoli attraverso l'uso di parentesi quadre;
- all'interno delle citazioni vanno indicate le omissioni, le quali possono essere segnalate attraverso l'uso dei puntini di sospensione racchiusi tra parentesi quadre (come per i commenti);
- quando si fanno delle omissioni nella citazione all'inizio o alla fine del brano, è preferibile l'uso dei puntini di sospensione senza che questi siano racchiusi tra parentesi quadre; all'inizio i puntini di sospensione sono staccati dalla prima parola, mentre alla fine sono attaccati all'ultima.

47.3.1.12 Acquisizione di termini stranieri

Quando si decide di utilizzare un termine straniero nel testo italiano, si pone il problema di stabilire il modo con cui questo possa convivere con il resto del testo. L'unica regola sicura è la verifica dell'uso generale. Tuttavia si possono definire alcune regole di massima, per dare l'idea del problema.

- La prima cosa da fare di fronte a un termine da non tradurre è di verificare in un vocabolario di lingua italiana; se c'è, il problema potrebbe essere considerato come risolto.
- Un termine straniero può assumere il genere che avrebbe se fosse tradotto in italiano, oppure quello che suona meglio dandogli un significato italiano. In caso di dubbio è importante controllare l'uso comune (se esiste).
- I termini stranieri non tradotti sono invariabili al plurale, cioè quando sono inseriti in testi in italiano vanno scritti sempre al singolare, senza aggiungere la lettera «s» finale, anche se ci si riferisce a una quantità maggiore di uno (si pensi al termine «mouse» che al plurale inglese diventa «mice»), perché la lingua italiana non può incorporare le regole di un'altra lingua.

Quando il termine che non si traduce non è di uso comune nell'ambiente a cui si rivolge il documento, dovrebbe essere evidenziato in corsivo tutte le volte che viene utilizzato. Per chiarire meglio il concetto, un termine tecnico può essere o meno di uso comune per il pubblico di lettori a cui si rivolge: se si tratta di un termine considerato normale per quell'ambiente, non è il caso di usare alcuna evidenziazione.

47.3.2 Anglofilia eccessiva

L'influenza della lingua inglese porta a deformazioni sempre più frequenti nella lingua italiana. Il problema più evidente, ma più facile da affrontare, è quello dei «falsi amici»: quei termini che, pur assomigliandosi (e pur avendo, spesso, la stessa etimologia), hanno significati diversi nelle due lingue. Gli esempi più celebri sono «factory» che diventa erroneamente «fattoria» e «cold» che si trasforma in «caldo».

Il problema meno evidente e per questo più insidioso è dato dalle altre differenze fra le due lingue: la punteggiatura, l'uso delle maiuscole e la struttura delle frasi. Trascurando queste particolarità si rischia di ottenere un testo che è formalmente in italiano, ma che non «suona» come tale.

Tabella 47.19 Traduzioni corrette dei «falsi amici».

consistent	coerente
exhaustive	esauriente
line	riga (quasi sempre)
re... (recursive)	ri... (ricorsivo)
set	insieme («set» è tennistico)
to set	impostare («settare» è di pessimo gusto)
subject	oggetto (di una lettera o di un messaggio)
to process	elaborare
to assume	supporre
proper (agg.)	giusto, corretto

proper (avv.)	vero e proprio
to support	si usi, per quanto possibile, una perifrasi
to return something	restituire qualcosa («ritornare» è intransitivo)

Quello che segue è un elenco di annotazioni riguardo all'uso dell'ortografia e della sintassi, influenzata erroneamente dalle consuetudini anglofone.

- La «e» o la «o» che introduce l'ultimo termine di un elenco non va preceduta da virgola. In inglese americano la norma è di usare la virgola (ma gli inglesi non la usano); a volte in italiano la virgola è ammissibile, ma si tratta di eccezioni.
- Se le frasi sono negative devono essere separate con «né». Per esempio, una frase del tipo «non possono essere cancellati o modificati» va espressa piuttosto come «non possono essere né cancellati, né modificati».
- I periodi italiani sono più complessi di quelli inglesi, a parità di registro, utilizzando opportunamente le congiunzioni, le subordinate, i due punti o i punti e virgola.
- I nomi dei mesi sono minuscoli.
- I numeri (interi) che esprimono una quantità piccola vanno scritti preferibilmente per esteso.
- In italiano si usa, di solito, la sequenza nome+aggettivo; il contrario, aggettivo+nome, per quanto accettabile, ha spesso un significato diverso. Per esempio, si osservi la differenza tra «pover'uomo» e «uomo povero».
- Bisogna sempre concordare il genere grammaticale: forme del tipo «la directory padre» non hanno senso.

47.3.3 Unità di misura

Nella documentazione a carattere scientifico diventa fondamentale la coerenza e la precisione nel modo in cui si indicano le grandezze e le unità di misura, oltre che la scelta di queste. In generale, ogni ambiente tecnico particolare tende a utilizzare le proprie grandezze e le proprie unità di misura, tralasciando gli sforzi di standardizzazione internazionale, contribuendo così a complicare inutilmente il proprio settore.

Purtroppo, l'ambito informatico costituisce l'esempio più problematico sotto questo aspetto, dal momento che l'esigenza di mantenere una compatibilità con il sistema binario ha attribuito a delle denominazioni ben precise del sistema decimale un significato differente rispetto a quello comune a tutti gli altri ambiti scientifici.

Lo standard internazionale sulle unità di misura è costituito dal SI, ovvero *Le Système international d'unités*, in italiano *Sistema internazionale di unità*. Il punto di riferimento per questo lavoro di armonizzazione è il BIPM (*Bureau international des poids et mesures*), con sede in Francia (*http://www.bipm.org/*).

47.3.3.1 Come si scrive una grandezza

Per esprimere una quantità riferita a una grandezza in modo grafico, occorre disporre del ***simbolo*** (la sigla) che ne esprime l'unità di misura o un multiplo opportuno di tale unità, al quale si fa precedere il numero, in cifre, di tale quantità:

n simbolo

È importante che tra il numero e la sigla ci sia uno spazio, il quale non deve poter essere interrotto in fase di impaginazione del testo. Per esempio: si può scrivere 5 kg, ma non 5kg.

47.3.3.2 Nomi e simboli

È bene chiarire il significato di alcuni termini che riguardano la misurazione di qualcosa:

Termine	Definizione
grandezza	ciò che viene misurato, come la lunghezza, la massa, il tempo;
unità di misura	il nome attribuito a ciò che si usa per misurare, come il metro, il kilogrammo, il secondo;
simbolo	il simbolo che rappresenta l'unità di misura in modo standard, come «m», «kg», «s».

Secondo il SI, il kilogrammo è l'unità di misura della massa, tenendo conto che i prefissi si utilizzano facendo riferimento al grammo. Si osservi inoltre che non si parla di «peso», perché questo termine è riferito piuttosto alla forza applicata a un oggetto.

I nomi delle unità di misura si esprimono generalmente senza iniziale maiuscola, mentre i simboli usati per rappresentarle simbolicamente vanno espressi esattamente come stabilito dagli standard, per quanto riguarda l'uso delle lettere maiuscole o minuscole.

Tabella 47.21. Esempi di grandezze e unità di misura.

Grandezza	Unità di misura	Simbolo
lunghezza	metro	m
massa	kilogrammo	kg
tempo	secondo	s
corrente elettrica	ampere	A

47.3.3.3 Prefissi moltiplicatori

Oltre alla definizione dei simboli che esprimono le unità di misura, si aggiungono dei simboli che rappresentano un multiplo ben preciso di tali unità. Tali simboli di moltiplicazione si pongono davanti al simbolo di unità a cui si riferiscono; per esempio, il simbolo «km» rappresenta mille unità «m», ovvero mille volte il metro.

I simboli che rappresentano tali moltiplicatori hanno anche un nome che normalmente si esprime senza iniziale maiuscola, indipendentemente dalla forma, maiuscola o minuscola, che ha il simbolo stesso.

I moltiplicatori riferiti alle unità di misura hanno un significato e un valore ben preciso. È un errore l'uso dei termini «kilo», «mega», «giga» e «tera», per rappresentare moltiplicatori pari a 2^{10}, 2^{20}, 2^{30} e 2^{40}, come si fa abitualmente per misurare grandezze riferite a bit o a byte.

Tabella 47.22. Prefissi del *Sistema internazionale di unità* (SI).

Nome	Simbolo	Valore	Note
yotta	Y	10^{24}	
zetta	Z	10^{21}	
exa	E	10^{18}	
peta	P	10^{15}	
tera	T	10^{12}	
giga	G	10^{9}	
mega	M	10^{6}	
kilo	k	10^{3}	Lettera «k» minuscola.
hecto, etto	h	10^{2}	
deca	da	10	
		1	Nessun moltiplicatore.
deci	d	10^{-1}	
centi	c	10^{-2}	
milli	m	10^{-3}	
micro	μ	10^{-6}	
nano	n	10^{-9}	
pico	p	10^{-12}	
femto	f	10^{-15}	
atto	a	10^{-18}	
zepto	z	10^{-21}	
yocto	y	10^{-24}	

47.3.3.4 Prefissi per multipli binari

Lo standard IEC 60027-2 introduce un gruppo nuovo di prefissi da utilizzare in alternativa a quelli del SI, per risolvere il problema dell'ambiguità causata dall'uso improprio dei prefissi del SI in ambito informatico. A questo proposito, una discussione particolareggiata su questo argomento si può trovare nel documento *Standardized Units for Use in Information Technology*, di Markus Kuhn, *http://www.cl.cam.ac.uk/~mgk25/information-units.txt* . La tabella 47.23 riporta l'elenco di questi prefissi speciali.

Tabella 47.23. Prefissi IEC 60027-2.

Origine	Nome	Simbolo	Valore	Note
kilobinary	kibi	Ki	2^{10}	Si usa la «K» maiuscola.
megabinary	mebi	Mi	2^{20}	
gigabinary	gibi	Gi	2^{30}	
terabinary	tebi	Ti	2^{40}	
petabinary	pebi	Pi	2^{50}	
exabinary	exbi	Ei	2^{60}	
zettabinary	zebi	Zi	2^{70}	
yottabinary	yobi	Yi	2^{80}	

47.3.4 Rappresentazione di valori

La rappresentazione di valori numerici tende a seguire forme differenti a seconda del contesto e delle convenzioni nazionali. Nella *Guide for the Use of the International Systems of Units (SI)*, pubblicato dal NIST (*National institute of standards and technology*), si trovano alcuni criteri per risolvere il problema in modo non ambiguo, validi anche al di fuori della realtà inglese.

47.3.4.1 Valori percentuali

In generale, l'uso del simbolo '**%**' va inteso come una forma abbreviata per 0,01 e in questo modo va usato, senza eccedere. In particolare, il simbolo di percentuale va posto dopo un valore numerico, staccato da questo, ma non separabile in fase di composizione tipografica:

```
n %
```

Per esempio, si può scrivere '`x = 0,025 = 2,5 %`', mentre non è corretta la forma '`x = 0,025 = 2,5%`'.

47.3.4.2 Valori numerici

Nella lingua italiana, come in molte altre, si usa la virgola come segno di separazione tra la parte intera e quella decimale, mentre nei paesi di lingua inglese, si utilizza il punto. A parte il problema di scegliere il segno opportuno in base alle proprie convenzioni nazionali, si pone piuttosto la difficoltà nel rappresentare numeri composti da una grande quantità di cifre.

La *Guide for the Use of the International Systems of Units (SI)* indica un metodo molto semplice e non equivoco: si separano le cifre a gruppi di tre, usando semplicemente uno spazio, sia prima, sia dopo il marcatore decimale, come si vede in questi esempi:

```
123 456 789
  3 456 789,012 345 6
      6 789,012 3
```

Naturalmente, lo spazio in questione non può consentire l'interruzione della riga in fase di composizione.

È ammissibile anche un'eccezione in presenza di raggruppamenti di sole quattro cifre, prima o dopo il marcatore decimale. In quel caso si può evitare la separazione:

```
1234
  23,2345
```

Un altro problema è quello della rappresentazione di valori numerici espressi con una base maggiore di 10, per i quali si utilizzano le prime 10 cifre numeriche e per il resto si usano le lettere alfabetiche.

Queste lettere andrebbero utilizzate coerentemente, possibilmente in forma maiuscola.

47.3.5 Stile tipografico

La definizione dello stile tipografico è un altro punto delicato nella definizione dello stile letterario generale. Di solito, la sua preparazione, è compito del tipografo o del coordinatore di un gruppo di autori o traduttori.

Il modo migliore per stabilire e utilizzare uno stile tipografico è quello di usare un sistema SGML, attraverso cui definire un DTD che non permetta alcun dubbio nella relazione che ci deve essere tra le varie componenti di un documento. In questo modo, gli autori hanno solo il compito di qualificare correttamente le varie componenti del testo, senza pensare al risultato finale, per modificare il quale si può semmai intervenire sul sistema di conversione successivo.

Le sezioni seguenti trattano dei problemi legati alla definizione di uno stile tipografico per la redazione di documenti tecnico-informatici. L'idea è presa dalla guida di stile del gruppo di documentazione di Linux: LDP (*Linux documentation project*), ma le indicazioni si basano sulle consuetudini tipografiche italiane.

47.3.5.1 Blocchi di testo letterale

Scrivendo documenti che riguardano l'uso dell'elaboratore, si incorre frequentemente nella necessità di scrivere nomi, o intere parti di testo, che devono essere trattati in modo letterale. Possono essere nomi di file e directory, comandi, porzioni del contenuto di file, listati di programmi, ecc. In questi casi è sconsigliabile l'uso di un tipo di carattere proporzionale, perché si rischierebbe di perdere delle informazioni importanti. Si pensi al trattino utilizzato nelle opzioni della maggior parte dei comandi Unix: utilizzando un carattere proporzionale, attraverso un sistema di composizione come LaTeX, si otterrebbe un trattino corto, mentre due trattini posti di seguito genererebbero un trattino normale; e ancora, da tre trattini si otterrebbe un trattino largo.

47.3.5.2 Nomi di file e directory

I nomi di file, di qualunque tipo, dovrebbero essere rappresentati attraverso un tipo di carattere a spaziatura fissa.

I nomi di questi tipi di entità sono sensibili alla differenza tra maiuscole e minuscole. Per questo vanno scritti sempre così come sono, anche quando si trovano all'inizio di un periodo, senza acquisire un'eventuale iniziale maiuscola.

I nomi di file eseguibili, in quanto tali, sono indicati preferibilmente senza il percorso necessario al loro avvio, ammesso che il sistema operativo non ne abbia bisogno per consentirne l'avvio.

I nomi di programmi per i sistemi Dos dovrebbero essere indicati utilizzando lettere maiuscole, senza tralasciare l'estensione.

47.3.5.3 Schermate, listati e simili

Il testo ottenuto da listati di vario tipo, come i pezzi di un programma sorgente, il risultato dell'elaborazione di un comando, o il contenuto di una schermata, possono essere rappresentati convenientemente attraverso un ambiente di inclusione di testo letterale a spaziatura fissa.

Il problema sta nel fatto che l'ampiezza di tale testo non può superare i margini del corpo del documento, in base al tipo di impaginazione finale che si ritiene dover applicare. Infatti, tale testo non può essere continuato nella riga successiva perché ciò costituirebbe un'alterazione delle informazioni che si vogliono mostrare.

Generalmente, non è possibile superare un'ampiezza di 80 colonne, pari a quella di uno schermo a caratteri tradizionale.

47.3.5.4 Variabili di ambiente

I nomi di variabili di ambiente dovrebbero essere rappresentati attraverso un tipo di carattere a spaziatura fissa.

I nomi di questi tipi di entità sono sensibili alla differenza tra maiuscole e minuscole. Per questo vanno scritti sempre così come sono, anche quando si trovano all'inizio o all'interno di un periodo.

A seconda del tipo di documentazione, potrebbe essere stata definita la convenzione per cui questi nomi debbano essere indicati sempre preceduti dal simbolo dollaro ('`$`').

La scelta di rappresentare le variabili utilizzando il dollaro come prefisso è motivata dalla facilità con cui queste possono essere poi identificate durante la lettura del testo. Tuttavia, una scelta di questo tipo potrebbe essere discutibile, perché il dollaro non appartiene al nome della variabile e perché potrebbe indurre il lettore a utilizzarlo sempre, anche quando negli script non si deve. Quindi, il buon senso deve guidare nella decisione finale.

47.3.5.5 Comandi e istruzioni

A volte si ha la necessità di indicare un comando, o un'istruzione, all'interno del testo normale. Per questo, è opportuno utilizzare un carattere a spaziatura fissa, come nel caso dei nomi di file e directory, però qui si pone un problema nuovo dovuto alla possibile presenza di spazi e trattini. I programmi di composizione normali tendono a interrompere le righe, quando necessario, in corrispondenza degli spazi ed eventualmente anche dei trattini. Se il comando o l'istruzione che si scrive è breve, è consigliabile l'utilizzo di spazi e trattini non interrompibili.[10]

Quando si utilizza SGML (compreso HTML), si può usare l'entità '` `' per indicare uno spazio non interrompibile, mentre se si usa solo LaTeX, è il carattere tilde ('`~`') che ha questa funzione.

Il problema del trattino non è semplice, perché non esiste un trattino generico non separabile, fine a se stesso. Di trattini ne esistono di varie misure e non sempre esistono corrispondenti per diversi tipi di programmi di composizione.

47.3.5.6 Nomi di applicativi

Quando si fa riferimento al nome di un programma si pongono due alternative: l'indicazione del file eseguibile oppure del nome attribuito dall'autore al suo applicativo.

Per comprendere la differenza, si può pensare a Apache: il servente HTTP. Non si tratta di un semplice eseguibile, ma di un applicativo composto da diverse parti, in cui l'eseguibile è '`httpd`'. Nello stesso modo, nel caso di Perl (il linguaggio di programmazione), si può pensare all'applicativo in generale, composto dalle librerie e tutto ciò che serve al suo funzionamento; oppure si può voler fare riferimento solo all'eseguibile: '`perl`'.

I nomi di programmi applicativi dovrebbero essere indicati nello stesso modo in cui lo fa il loro autore, rispettando l'uso delle maiuscole e delle minuscole, in qualunque posizione del testo.

I nomi di questi tipi di entità non dovrebbero essere evidenziati in modo particolare.

47.3.5.7 Concetti e termini nuovi

I concetti e i termini che non si ritengono familiari per il lettore, dovrebbero essere evidenziati la prima volta che si presentano.

Per questo tipo di evidenziazione si utilizza un neretto oppure un corsivo. L'uso del neretto è contrario alla tradizione dei testi italiani, in cui questo viene fatto normalmente utilizzando solo il corsivo. Tuttavia, il neretto si presta meglio alla composizione in formati molto diversi; per esempio si ottiene facilmente anche su un documento da visualizzare attraverso uno schermo a caratteri.

47.3.5.8 Nomi proprietari e logotipi

L'indicazione di nomi che fanno riferimento a marchi di fabbrica o simili, va fatta come appare nel copyright o nella nota che fa riferimento al brevetto, rispettando l'uso delle maiuscole e dell'eventuale punteggiatura. Si dovrebbe evitare, quindi, di prendere in considerazione un eventuale logo grafico del prodotto, in quanto non è il caso di fare risaltare maggiormente i nomi di questo tipo. Ma a questa regola si può aggiungere che, nel caso il nome sia scritto utilizzando solo lettere maiuscole, può essere opportuno limitarsi a indicarlo utilizzando solo l'iniziale maiuscola, lasciando il resto in minuscolo.

All'interno del testo non è conveniente fare riferimento a detentori di copyright o di brevetti; eventualmente, di questo problema dovrebbero farsi carico delle note opportune all'inizio del documento che si scrive (ma solo se lo si ravvisa necessario per qualche ragione particolare). A tale proposito si osservi che in generale non è indispensabile fare alcun tipo di riferimento di questo genere, se lo scopo di ciò che si scrive non è quello di trattare espressamente di questo o di quel prodotto.

47.3.5.9 Titoli

Nei testi di lingua italiana, i titoli vanno scritti come se si trattasse di testo normale, con le particolarità seguenti:

- non viene mai posto il punto fermo finale;
- si cerca di evitare l'inserzione di altri segni di punteggiatura, a meno che ciò sia necessario per qualche motivo;
- non si usano evidenziazioni particolari di parole o nomi come invece potrebbe avvenire nel testo normale.

Un documento a carattere tecnico viene suddiviso normalmente in segmenti a più livelli, ma in generale è bene limitare la profondità di questi, per quanto possibile.

47.3.5.10 Didascalie

Gli elementi che non fanno parte del flusso normale di un documento, come tabelle e figure, sono accompagnate generalmente da un titolo e da una didascalia. Il titolo serve a identificarle, mentre la didascalia ne descrive il contenuto.

I titoli di tabelle, figure e oggetti simili, seguono le regole dei titoli normali, mentre il testo delle didascalie segue le regole del testo normale. Tuttavia, quando si utilizzano programmi di composizione che permettono di abbinare solo una nota descrittiva, che funga sia da titolo, sia da didascalia, occorre fare una scelta:

- quando le note sono brevi, è opportuno che si comportino come i titoli, cioè non contengano simboli di punteggiatura;
- quando sono più lunghe, si può decidere di trattarle come didascalie vere e proprie, con tutti i simboli di punteggiatura necessari per una comprensione corretta del contenuto.

Naturalmente, la scelta fatta deve valere per tutte le descrizioni che si abbinano a questi oggetti di un particolare documento: brevi o lunghe che siano.

47.3.5.11 Elenchi descrittivi

Gli elenchi descrittivi, come quelli che si ottengono con HTML utilizzando la struttura seguente, possono essere insidiosi, perché potrebbero tradursi in modo differente a seconda del tipo di programma di composizione utilizzato.

```
<dl>
<dt>Primo elemento</dt>
<dd>
    <p>Descrizione del primo elemento,...
    Bla bla bla...</p>
</dd>
</dl>
```


L'elemento descrittivo dell'elenco è in pratica un titolo che introduce una parte di testo generalmente rientrata. Sotto questo aspetto, la voce descrittiva segue le regole già viste per i titoli. Tuttavia, il problema sta nel fatto che si potrebbe essere indotti a riprendere un discorso lasciato in sospeso quando veniva introdotto l'elenco, come nell'esempio seguente:

```
Bla bla bla bla…

Primo elemento

        Descrizione del primo elemento,…
        Bla bla bla…

Qui si riprende il discorso precedente all'elenco
descrittivo.
…
```

Infatti, l'utilizzo dei rientri fa percepire immediatamente la conclusione dell'elenco stesso. Quando si scrive un documento che deve poter essere convertito in molti formati differenti, che quindi potrebbe essere elaborato da programmi di composizione di vario tipo, può darsi che i rientri vengano perduti e gli elementi descrittivi dell'elenco appaiano come dei titoli veri e propri. Ma se ciò accade, quando si ricomincia «il discorso lasciato in sospeso», sembra che questo appartenga all'argomento dell'ultimo titolo apparso.

```
Bla bla bla bla…

Primo elemento

Descrizione del primo elemento,…
Bla bla bla…

Qui si riprende il discorso precedente all'elenco
descrittivo.
…
```

Pertanto, se si vogliono utilizzare strutture di questo tipo, è consigliabile che appaiano alla fine di una sezione, quando quello che viene dopo è un titolo di una sezione o di qualcosa di simile.

47.3.5.12 Richiami in nota

I richiami in nota (le note a piè pagina e quelle alla fine del documento) sono composti con le stesse regole del testo normale. Quando il riferimento a una nota si trova alla fine di una parola cui segue un segno di interpunzione, è opportuno collocare tale riferimento dopo il simbolo di interpunzione stesso.

47.3.5.13 Indicizzazione

La costruzione di un indice analitico deriva dall'inserzione di riferimenti all'interno del testo, attraverso istruzioni opportune definite dal tipo di programma usato per la composizione.

Le voci inserite in questi riferimenti, che poi vanno a formare l'indice analitico, devono essere scelte in modo uniforme, secondo alcune regole molto semplici.

- Si utilizzano le lettere minuscole, a meno che si tratti di nomi particolari che vanno sempre scritti in un modo prestabilito:
  - i nomi proprietari vanno scritti come indicato dalla casa produttrice;
  - i nomi di applicativi software vanno scritti come indicato dall'autore;
  - i nomi di file e directory vanno scritti esattamente come sono, tenendo conto che i file eseguibili vanno indicati senza percorso, mentre gli altri potrebbero contenerlo;
  - i nomi di variabili di ambiente vanno scritti esattamente come sono, eventualmente prefissati dal simbolo dollaro.
- Si utilizza solo il singolare.

I riferimenti per la generazione dell'indice analitico vanno posti preferibilmente nei luoghi opportuni, in modo da evitare inutili rimandi

a pagine che non contengono ciò che si cerca. Per esempio, la parola **file** potrebbe trovarsi in quasi tutte le pagine di un testo di informatica, mentre è conveniente che l'indice analitico riporti solo le pagine in cui si parla del concetto che questa parola rappresenta.

I nomi di programmi eseguibili e di file di dati standard, come per esempio i file di configurazione, dovrebbero essere inseriti nell'indice analitico ogni volta che appaiono nel testo.

47.3.5.14 Riferimenti bibliografici e simili

Esiste una forma precisa e molto articolata per la stesura delle bibliografie, corrispondente allo standard ISO 690. A ogni modo, vale la regola generale per cui un riferimento bibliografico deve contenere tutti i dati necessari a reperire il documento a cui si fa riferimento. In condizioni normali, le informazioni essenziali per identificare una pubblicazione sono quelle seguenti:

- l'autore o gli autori;
- il titolo completo;
- l'editore;
- la data di edizione;
- il numero ISBN (se disponibile);
- l'URI (se il documento è disponibile attraverso la rete).

Generalmente è consigliabile comporre gli elenchi bibliografici indicando le opere a partire dall'autore, mettendo il titolo in testo corsivo o inclinato, separando le varie componenti di ogni riferimento bibliografico attraverso delle virgole, come nell'esempio seguente:

> Claudio Beccari, *LaTeX, Guida a un sistema di editoria elettronica*, Hoepli, 1991, ISBN 88-203-1931-4

Se non si dispone di un sistema automatico per la gestione dei riferimenti bibliografici, quando si cita un documento all'interno del testo, è bene seguire alcune regole elementari.

- I riferimenti ad altri documenti, all'interno del testo normale, vanno fatti indicando il titolo completo, in corsivo o inclinato, aggiungendo il nome dell'autore o degli autori.
- Il titolo è separato con una virgola da un eventuale sottotitolo.
- I riferimenti a un testo già citato possono essere fatti utilizzando solo il titolo o solo l'autore, o attraverso altri mezzi, purché si sia certi di non creare ambiguità o disagio al lettore.

Segue un esempio molto semplice di come può essere fatto un riferimento del genere all'interno del testo normale:

> ... Questa sezione fa riferimento a concetti contenuti in *LaTeX, Guida a un sistema di editoria elettronica*, di Claudio Beccari...

47.4 Evoluzione dell'editoria elettronica

Con il termine «editoria elettronica», si vuole fare riferimento agli strumenti utilizzabili per produrre documentazione di buona qualità dal punto di vista tipografico. L'approccio di un programma per l'editoria può essere fondamentalmente di due tipi:

- a composizione visuale o WYSIWYG (*What you see is what you get*);[11]
- a composizione differita.

Nel primo caso, durante la stesura, il documento appare sullo schermo con lo stesso aspetto che avrebbe se venisse stampato in quel momento. Nel secondo, si scrive un file di testo normale con l'inserimento di comandi, come se si trattasse di un linguaggio di programmazione; quindi si passa alla composizione (una sorta di compilazione) attraverso la quale viene generato normalmente il file finale pronto per essere inviato alla stampa.

Il primo tipo di composizione è decisamente più pesante sotto l'aspetto elaborativo, prestandosi in particolare per i documenti brevi. Il secondo ha lo svantaggio di non permettere la verifica del risultato finale fino a quando non avviene la composizione, però richiede solo l'utilizzo di un programma normalissimo per la creazione e la modifica di file di testo, mentre solo al momento della composizione c'è bisogno di un'elaborazione significativa. In questo senso è più adatto alla redazione di documenti di grandi dimensioni.

47.4.1 Evoluzione

L'editoria elettronica non è più solo cartacea. In particolare esistono gli ipertesti, cioè documenti elettronici la cui consultazione avviene attraverso riferimenti e non in modo puramente sequenziale.

In questo senso, se l'editoria elettronica viene vista come mezzo di documentazione generale non più orientata a un supporto particolare, non può avere immediatamente una rappresentazione finale definitiva. Per esempio, un documento in HTML non può mai essere identico a un documento stampato.

Quando si vuole produrre un documento compatibile con diversi tipi di supporti (carta, ipertesto HTML, guida interna, ecc.) non si possono avere pretese stilistiche particolari; quindi, un programma visuale diventa quasi inutile.

A fianco di questi problemi di compatibilità, si aggiungono delle esigenze nuove, come per esempio la possibilità di estrarre dal documento elettronico determinati tipi di informazioni necessarie ad alimentare una base di dati. In questo senso, le informazioni cercate, oltre che riconoscibili all'interno del formato utilizzato, devono essere coerenti e complete.

Comunque, anche nell'ambito dell'editoria cartacea tradizionale, la prima esigenza che è stata sentita è quella dell'uniformità stilistica, cosa che sarebbe bene fosse controllabile anche attraverso il sistema elettronico di composizione.

47.4.2 Codifica del testo (markup)

Il termine *markup* (o marcatura) deriva dall'ambiente tipografico dove è stato usato per definire le annotazioni fatte su una bozza, allo scopo di segnalare al compositore o al dattilografo il modo con cui alcune parti del testo andavano evidenziate o corrette. A tale proposito, esiste uno standard nella simbologia da utilizzare in questi casi, rintracciabile ancora nei libri di tipografia. Queste annotazioni simboliche possono riferirsi all'aspetto dei caratteri, all'allineamento dei paragrafi, alle spaziature e via dicendo.

Nell'editoria elettronica, il concetto alla base del termine *markup* si è esteso in modo da includere i simboli speciali, o meglio, la codifica inserita nel testo per permetterne l'elaborazione.

Volendo generalizzare, la codifica del testo è tutto ciò che ne esplicita l'interpretazione. A livello umano, la stessa punteggiatura e certe forme di spaziatura, sono la codifica che serve a chiarire il significato del testo, diventando parte essenziale di questo. Oggi non sarebbe comprensibile separare concettualmente la punteggiatura dal testo, però in passato è stato così. Basta pensare ai telegrammi, o all'apparizione di questi simboli nella storia della scrittura.

47.4.2.1 Linguaggio di markup

La tecnica di composizione del testo utilizzando l'inserimento di marcatori o di codici, richiede la definizione di una serie di convenzioni, tali da definire un ***linguaggio di markup***. Un tale linguaggio deve specificare quale tipo di marcatura è utilizzabile, quale è richiesta, in che modo si distingua dal testo e quale sia il suo significato.

I linguaggi di *markup* possono essere diversi e si distinguono due gruppi fondamentali: linguaggi procedurali e linguaggi descrittivi.

Un linguaggio di *markup* procedurale serve a definire il processo da svolgere in un punto particolare del documento. È come un linguaggio di programmazione in cui si usano chiamate di funzioni, o

di procedure, per compiere le operazioni richieste. Per esempio può trattarsi di ordini riferiti alla scrittura del testo, allo spostamento, alla definizione di margini, del salto pagina e di tutto ciò che si rende necessario. In questo senso, un linguaggio di *markup* procedurale consente generalmente la definizione completa di tutto ciò che serve a stabilire l'aspetto finale del documento stampato (o visualizzato).

Un linguaggio di *markup* descrittivo, al contrario, usa la codifica dei marcatori per classificare le parti del documento, dando loro un nome. In pratica, si delimitano queste porzioni di testo e si definisce la loro appartenenza a una categoria determinata, identificata da un nome. In tal modo, questo tipo di linguaggio di *markup* non è in grado di fornire indicazioni sull'aspetto finale del documento, in quanto il suo scopo è solo quello di definire la struttura del testo. Evidentemente è compito di un'altra applicazione utilizzare le informazioni sulla struttura del testo per generare un formato finale, secondo regole e definizioni stabilite al di fuori del linguaggio descrittivo stesso.

47.4.2.2 Vantaggi di un linguaggio descrittivo

Un linguaggio di *markup* descrittivo, nel momento in cui non si prende carico di definire l'aspetto finale del documento, pone l'accento sul contenuto e non sull'apparenza. Questo è fondamentale quando il «documento» viene inteso come informazione pura che possa materializzarsi in forme molto diverse.

L'informazione «pura», in quanto tale, richiede anche che sia espressa attraverso un formato indipendente dalle piattaforme, ma soprattutto che sia indipendente dai formati proprietari.

47.4.3 SGML

SGML è un linguaggio di *markup* descrittivo, definito dallo standard *ISO 8879: Information processing---Text and office systems---Standard Generalized Markup Language (SGML)*, 1986. L'SGML è uno standard internazionale per la definizione di metodi di rappresentazione del testo in forma elettronica in modo indipendente dall'hardware e dal sistema utilizzato.

Il linguaggio SGML utilizza il concetto di «tipo di documento» e di «definizione del tipo di documento». Per la precisione di parla di DTD, ovvero, *Document type definition*. In pratica, nell'ambito dell'SGML, è necessario che sia stato definito il modo in cui i vari elementi del testo possono essere utilizzati. Ciò che non è definito, non può essere usato, ma quello che è stato definito deve rispettare le regole stabilite.

A titolo di esempio, si può immaginare la definizione di un tipo di documento riferito alla scrittura di lettere commerciali. La lettera deve contenere degli elementi essenziali: il mittente, uno o più destinatari, la data, l'oggetto, il corpo, l'indicazione di colui che la firma e la sigla del dattilografo che la scrive materialmente. Tutti questi elementi devono essere presenti, probabilmente anche con un certo ordine (l'indicazione di chi firma deve trovarsi in fondo e non all'inizio). Inoltre, questi elementi possono scomporsi in altri elementi più dettagliati; per esempio, l'informazione sulla persona che firma può comporsi della qualifica, il titolo personale, il nome e il cognome. Il DTD deve prendersi carico di definire tutto questo, stabilendo ciò che è valido e cosa invece non lo è.

In questo modo, poi, un documento SGML può essere analizzato da un programma speciale, l'analizzatore SGML (*SGML parser*), per la verifica del rispetto di queste regole, prima di utilizzare in qualunque modo questo documento.

L'SGML, assieme al DTD, garantendo l'uniformità dei documenti dello stesso tipo, consente di uniformare i procedimenti successivi. Per tornare all'esempio precedente, da un punto di vista di puro contenuto del testo, non dovrebbe essere importante l'ordine degli elementi che lo compongono, quando sia possibile distinguerli. Tuttavia, una lettera che inizia con la firma e finisce con l'indicazione del destinatario, non è scritta nel modo corretto; così il DTD potrebbe essere progettato in modo da imporre un certo ordine, a vantaggio delle elaborazioni successive.

47.4.4 XML, XSLT e XSL-FO

XML (*Extensible markup language*) è un linguaggio derivato da SGML, nato inizialmente come sottoinsieme «compatibile» con questo. A XML si affianca XSLT (*Extensible stylesheet language transformations*), come linguaggio di definizione della trasformazione di un documento XML in qualcosa di differente. XSL-FO (*Extensible stylesheet language formatting object*) è un linguaggio di definizione del formato finale del documento; spesso si usa XSLT per ottenere un documento XSL-FO, per poi passare alla composizione finale.

47.5 Lettera, codifica e carattere da stampa

La codifica dei caratteri, intendendo ciò come il modo di rappresentare i simboli tipografici in forma elettronica, diventa un problema serio nel momento in cui si esce dallo schema abituale della scrittura con caratteri latini.

Nella storia dell'informatica è stata definita una quantità enorme di codifiche differenti, per adattare la limitatezza degli 8 bit tradizionali all'insieme di caratteri che serve in ogni circostanza particolare. Inoltre, nelle situazioni in cui 8 bit non possono bastare, sono state ideate codifiche attraverso l'abbinamento di sequenze di simboli elementari che ne rappresentano uno più complesso.

In generale, verrebbe da pensare che sarebbe stato meglio prevedere subito il problema, definendo delle unità di codifica più grandi (non più il byte di 8 bit, ma stringhe binarie più lunghe). Tuttavia, c'è da considerare che proprio la semplicità dell'alfabeto inglese (che non ha nemmeno le lettere accentate) ha permesso lo sviluppo rapido di tecnologia relativamente «semplice», che altrimenti sarebbe stata materialmente irraggiungibile.

Il byte stesso (quello da 8 bit) è stata una grande conquista. Ancora oggi ci sono sistemi di comunicazione che riconoscono unità di codifica a soli 7 bit, dove in pratica si può usare solo l'ASCII; prima ancora sono state utilizzate anche unità di codifica a soli 6 bit.

Per comprendere il problema della codifica, è necessario considerare i problemi che riguardano la definizione dei caratteri da stampa. La prima fase è la definizione di un repertorio astratto, all'interno del quale si elencano, senza un ordine preciso, le lettere e gli altri segni necessari per un certo scopo. L'insieme di questi simboli è astratto, nel senso che non è ancora stabilito l'aspetto finale, compito riservato a una fase successiva.[12]

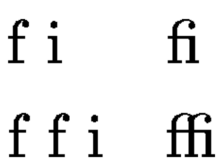
Il simbolo di un repertorio astratto è qualcosa di diverso dal simbolo che compone un carattere da stampa, dal momento che il secondo rappresenta il modo preciso in cui il simbolo astratto viene reso tipograficamente. Per comprendere il concetto, si pensi alla lettera «a» e all'aspetto che questo simbolo astratto può avere utilizzando stili, serie e corpi differenti. Evidentemente, si tratta sempre della stessa lettera, ma resa in modo diverso.

Figura 47.29. La lettera «a» minuscola resa tipograficamente in modo differente.

Alcuni gruppi di simboli astratti tendono a essere rappresentati tipograficamente in un simbolo solo, in un legato, ovvero attraverso l'avvicinamento e la sovrapposizione parziale. Il caso tipico è rappresentato dalla sequenza di lettere «fi» e «ffi», come si vede nella figura 47.30. In certi casi, la sequenza di lettere che si avvicina rappresenta una parola intera, generando così un «logotipo»; spesso, la loro importanza storica ha fatto sì che questi siano diventati dei simboli astratti autonomi. Per esempio, la parola latina «et» è diventata la e-commerciale odierna, «&»; la parola latina «ad» è diventata la

chiocciola odierna, «@» (nella lingua italiana si è perso l'uso di questo legato, riacquisendolo solo attraverso la lingua inglese, pertanto lo si conosce solitamente solo nella sua definizione inglese: *at*).

Figura 47.30. Il legato «fi» e «ffi».



La composizione tipografica elettronica, può avvenire attraverso la sovrapposizione di simboli elementari differenti, senza la necessità di legarli assieme. Nelle lingue di origine latina, il caso più comune di questa possibilità si ha con gli accenti, i quali potrebbero essere simboli tipografici separati da sovrapporre alle lettere a cui sono destinati. Nello stesso modo, il repertorio di simboli astratti potrebbe essere realizzato con questo criterio; per esempio, per fare riferimento a un simbolo complesso potrebbe essere necessario indicare una sequenza di simboli astratti elementari.

Alcune lingue hanno dei simboli che nella composizione tipografica devono cambiare forma a seconda del contesto. Per comprendere il concetto, si può pensare a una scrittura manuale, in cui le lettere cambiano leggermente forma a seconda di ciò che appare prima e dopo; chi conosce una scrittura stenografica manuale, può intendere ancora meglio il problema. Ad aggravare ancora di più la questione, l'adattamento contestuale di un simbolo potrebbe dipendere da una scelta stilistica, in parte arbitraria.

A volte, la larghezza di un testo deve essere adattata per esigenze estetiche, come avviene nel caso dell'allineamento simultaneo a sinistra e a destra. Nelle lingue di origine latina si ottiene questo attraverso l'allargamento degli spazi tra le parole e tra le lettere all'interno delle parole; tuttavia, alcune lingue richiedono degli adattamenti differenti, per esempio attraverso l'introduzione di altri simboli appropriati.

Da quello che è stato scritto si intende che la composizione tipografica elettronica si può considerare come l'ultima fase di un processo composto da tre livelli: definizione di un repertorio astratto di simboli; definizione di una codifica; composizione tipografica a partire dalla codifica.

repertorio astratto ⟶ codifica ⟶ composizione

La codifica non può corrispondere esattamente al repertorio astratto ideale: deve fare delle scelte. In generale, il repertorio simbolico preso in considerazione dalla codifica è identificabile come un insieme di ***punti di codifica*** (*code point*, secondo la documentazione di Unicode).

I problemi legati alla composizione tipografica che sono stati descritti, sono solo alcuni di quelli che si incontrano. A seconda dei casi, implicano un approccio differente per ciò che riguarda la codifica e la composizione. In breve:

1. lo stile tipografico è qualcosa che normalmente è gestito dal sistema di composizione, senza richiedere la definizione di punti di codifica differenti;
2. il legato può essere un problema risolto a livello di composizione finale, oppure può richiedere la definizione di punti di codifica aggiuntivi, quando si tratta di legati molto importanti o di logotipi;
3. l'adattamento contestuale richiede spesso la definizione di tanti punti di codifica quante sono le varianti contestuali del simbolo astratto, specialmente se esiste un margine di scelta da parte dell'autore;
4. l'adattamento della larghezza del testo dovrebbe essere compito del sistema di composizione, anche quando questo implica l'inserzione di simboli speciali.

47.6 Ambiguità nel concetto di «carattere»

In informatica, il termine «carattere» ha acquisito un significato ambiguo che dipende dal contesto. Per esempio, può riferirsi a un simbolo del repertorio astratto, a un punto di codifica, all'unità di memorizzazione (unità di codifica, o *code unit*), o al segno che viene ottenuto alla fine del processo di composizione. Per fare un esempio in merito all'unità di codifica, basta pensare al byte che spesso viene confuso con il carattere, mentre ormai è da intendersi come un'unità di memorizzazione troppo piccola per questo scopo nel sistema globale.

Di certo non si può pretendere che si smetta di usare il termine carattere per passare invece a una terminologia più precisa. Tuttavia è importante rendersi conto della vastità della cosa e dei problemi che ci stanno sotto.

Il modello di Unicode suddivide il problema della codifica in cinque livelli:

1. ACR (*Abstract character repertoire*)
   definizione di un repertorio astratto di simboli;
2. CCS (*Coded character set*)
   definizione di una mappa in cui si abbina un numero intero, non negativo, a ogni simbolo del repertorio astratto che si intende gestire;
3. CEF (*Character encoding form*)
   definizione di una mappa in cui si abbinano i numeri ottenuti dal livello precedente a un insieme di sequenze dell'unità di codifica prescelta;
4. CES (*Character encoding scheme*)
   definizione di una mappa di trasformazione delle unità di codifica in una sequenza seriale di byte;
5. TES (*Transfer encoding syntax*)
   definizione di un metodo reversibile di trasformazione dei dati codificati in base alle limitazioni del mezzo trasmissivo.

Da un punto di vista leggermente differente, si potrebbe scomporre il problema in strati, per distinguere le fasi che vanno dalla scrittura alla trasmissione del testo e dalla ricezione del testo alla lettura. La scrittura potrebbe essere descritta con l'elenco di operazioni seguenti:

1. selezioni dei simboli e digitazione attraverso la tastiera (o un altro mezzo);
2. codifica, attraverso cui si trasforma il simbolo in un numero intero non negativo;
3. trasformazione in unità di codifica, in base alla forma prescelta;
4. adattamento in sequenze di byte;
5. adattamento prima del trasferimento dei dati.

Il processo di lettura dei dati, a partire dalla ricezione, è opposto:

1. interpretazione dei dati ricevuti e ricostruzione delle sequenze di byte di partenza;
2. trasformazione delle sequenze di byte in sequenze di unità di codifica;
3. trasformazione dalle sequenze di unità di codifica in numeri interi non negativi;
4. decodifica dei numeri interi non negativi;
5. composizione tipografica (su schermo o su carta).

Prima di questa sezione è già stato affrontato il problema dell'abbinamento tra il repertorio astratto di simboli e la codifica, senza precisare in che modo sia organizzata questa ultima. Nelle sezioni seguenti si accenna alle problematiche successive.

47.6.1 CCS: insieme di caratteri codificato

L'insieme di caratteri codificato è in pratica il repertorio simbolico disponibile effettivamente, ottenuto dopo la definizione di un repertorio astratto e dopo lo studio dei problemi legati alla cultura e alle consuetudini del linguaggio per il quale è stato realizzato. Questo insieme di caratteri abbina un numero intero a ogni simbolo, senza bisogno che ci sia continuità nella sequenza di tale valore (l'unica limitazione è quella per cui deve trattarsi di un valore non negativo).

Il numero che rappresenta il simbolo di un insieme di caratteri codificato, è il punto di codifica.

Nella documentazione tecnica si fa spesso riferimento al concetto di «insieme di caratteri», ovvero *character set*, per intendere quello che qui si indica come «insieme di caratteri codificato», ovvero CCS, *Coded character set*.

Alcuni esempi tradizionali di insiemi di caratteri codificati sono:

ASCII (ISO 646)	127 punti di codifica;
ISO 8859-1	i primi 127 punti di codifica sono uguali all'ASCII;
ISO 8859-2	i primi 127 punti di codifica sono uguali all'ASCII, mentre nella parte restante il repertorio dei simboli è diverso dall'ISO 8859-1.
Unicode (ISO 10646)	i primi 255 punti di codifica sono uguali a ISO 8859-1.

Alcuni insiemi di caratteri codificati prevedono l'abbinamento con una descrizione (in inglese), allo scopo di facilitarne l'identificazione. Si utilizza questa descrizione per evitare ambiguità nell'identificazione del simbolo, quando questo potrebbe essere confuso con un altro, o più semplicemente quando potrebbe essere male interpretato.

Per rappresentare un punto di codifica, basta indicare il suo numero intero (qualunque sia la sua base di numerazione). Di solito, per evitare ambiguità, quando si tratta di Unicode o di ISO 10646, si fa uso normalmente della forma '**U+*nnnn***', oppure '**U-*nnnnnnnn***', dove ***n*** è una cifra esadecimale. Evidentemente, la seconda forma è utile per individuare punti di codifica più grandi. Per la precisione, questa notazione è la rappresentazione delle codifiche UCS-2 e UCS-4 a cui non si intende fare riferimento direttamente. In generale, non c'è alcun bisogno di rappresentare un punto di codifica in questo modo; tuttavia, si tratta di una simbologia immediata che dovrebbe semplificare la lettura e la comprensione del testo.

Tuttavia, in questo capitolo, per maggiore chiarezza, si preferisce l'uso di una forma differente, mediata dall'XML, '**#x*n***', dove ***n*** rappresenta una o più cifre esadecimali in base alla necessità.

In questa fase della scomposizione del problema della codifica, il «carattere» è il numero intero che rappresenta il punto di codifica. Attraverso un linguaggio di programmazione che sia adeguato al problema della codifica universale, il tipo di dati «carattere» deve corrispondere a un intero senza segno per il quale non ci si pone il problema del limite (anche se in questo momento dovrebbe essere almeno un intero a 32 bit); di conseguenza, il tipo stringa dovrebbe essere un array del tipo carattere.[13]

47.6.2 CEF: forma codificata del carattere

La forma codificata del carattere è il risultato di una trasformazione dal numero intero non negativo che costituisce il livello precedente, in una sequenza di unità di codifica. L'unità di codifica è un raggruppamento di bit di una lunghezza opportuna.

La sequenza di unità di codifica non è composta necessariamente dalla stessa quantità di queste unità per tutti gli elementi dell'insieme di caratteri. A tale proposito, si distingue tra forme codificate del carattere a lunghezza fissa e a lunghezza variabile.

L'esempio più semplice di forma codificata del carattere a lunghezza fissa è dato dall'ASCII tradizionale: l'insieme di caratteri codificato è costituito da 128 punti di codifica, rappresentati da tutti gli interi che vanno da 0 a 127. L'unità di codifica utilizzata in questa situazione è un gruppo singolo di 7 bit con i quali si rappresenta lo stesso numero intero.

Il caso più comune di forma codificata del carattere a lunghezza variabile è dato dall'UTF-8, che utilizza un'unità di codifica di un ottetto (un byte), in cui i punti di codifica con valori tra 0 e 127 (da 00_{16} a $7F_{16}$) utilizzano una sola unità di codifica, mentre tutti gli altri ne utilizzano più di una.

In fase di interpretazione delle sequenze di unità di codifica si possono presentare i casi seguenti:

1. la sequenza potrebbe non essere valida, perché incompleta, o perché esclusa esplicitamente;
2. la sequenza potrebbe fare riferimento a un punto di codifica possibile ma non ancora assegnato a un simbolo;
3. la sequenza potrebbe corrispondere a un punto di codifica assegnato a un simbolo stabilito, oppure lasciato all'attribuzione libera senza un vincolo preciso.

Il problema delle sequenze incomplete si intende nel momento in cui si accetta il fatto che una forma di codifica possa prevedere una lunghezza variabile delle sequenze di unità di codifica. Il caso dei punti di codifica lasciati al libero arbitrio degli utilizzatori, è una particolarità della codifica universale (Unicode e ISO 10646); se ne può comprendere la necessità di fronte a un sistema di codifica che vuole essere completo, ma che in pratica è appena all'inizio della sua opera di catalogazione.

A questo livello della scomposizione del problema, il «carattere» è ciò che idealmente è scritto in un «file di testo» (non più solo un «file ASCII»). Anche se è stato stabilito in che modo è organizzato l'insieme di caratteri codificato, la sua rappresentazione binaria «ideale» nel file di testo dipende dalla forma prescelta. Qui si parla di rappresentazione ideale, perché la rappresentazione reale dipende dal livello successivo, in cui tutto viene tradotto a livello di byte.

47.6.3 CES: schema di codifica del carattere

Lo schema di codifica del carattere è un sistema di trasformazione attraverso il quale, le unità di codifica vengono rese in sequenze di byte messe in serie.

Per tornare all'esempio dell'ASCII, l'unità di codifica è di 7 bit, ma il «carattere» ASCII si gestisce in pratica all'interno di un byte, dove il bit più significativo viene lasciato azzerato.

In generale, il byte è un'unità di memorizzazione standard in tutte le architetture dei sistemi elaborativi e in tutti i sistemi di trasmissione dati. Questo spiega la necessità di trasferire tutto a livello di byte o di multipli di questa unità.

Dovendo utilizzare più byte per rappresentare un oggetto unico, si pone il problema dello scambio tra coppie di byte che avviene in alcune architetture. Come è noto, si distingue tra *big-endian*, in cui il primo byte è quello più significativo, e *little-endian*, in cui il primo byte è quello meno significativo. Pertanto, in questa situazione, si impone la necessità di specificare l'ordine dei byte.

47.6.4 TES: sintassi di codifica per il trasferimento

La sintassi di codifica per il trasferimento è un metodo di trasformazione reversibile di una codifica, che si deve attuare a causa di qualche tipo di esigenza. Per esempio:

- la necessità di evitare l'utilizzo di alcuni valori nei byte che potrebbero confondere un sistema di comunicazione o di memorizzazione;
- la necessità di ridurre la dimensione dei dati utilizzando algoritmi di compressione.

Mentre il secondo caso dovrebbe essere chiaro, per comprendere il primo basta pensare alle limitazioni che ha storicamente il protocollo SMTP (posta elettronica), per cui è necessario evitare di trasmettere byte in cui il primo bit sia diverso da zero.

47.7 Codifica in pratica: da Unicode a ASCII

Il lavoro per la realizzazione dell'insieme di caratteri universale non può partire da zero, per l'esigenza di mantenere qualche forma di compatibilità con il passato (diversamente, non verrebbe nemmeno preso in considerazione). Pertanto, le incongruenze che si possono rilevare sono dovute principalmente a questo motivo: la necessità di riutilizzare gli insiemi di caratteri codificati più importanti già esistenti.

Unicode e ISO 10646 sono due standard compatibili reciprocamente che definiscono un insieme di caratteri codificato particolarmente grande, da trasformare poi nella forma codificata del carattere prescelta per la sua rappresentazione pratica in unità di codifica. Pertanto, quando di parla di Unicode, o di ISO 10646, senza specificare altro, si pensa generalmente ai punti di codifica e non alla rappresentazione finale.[14]

I primi punti di codifica di questi standard corrispondono esattamente all'ISO 8859-1. Per esempio: #x20 è lo spazio normale; #xA0 è lo spazio non interrompibile; #xAB sono le virgolette angolari aperte; #xBB sono le virgolette angolari chiuse.

Attualmente, l'insieme di caratteri universale utilizza principalmente tre forme codificate del carattere UTF (*Unicode transformation format*): UTF-8, UTF-16 e UTF-32. Ogni forma codificata del carattere del tipo UTF-***n*** rappresenta un punto di codifica come una sequenza di una o più unità di codifica (che a sua volta occupa ***n*** bit), ottenuta attraverso una trasformazione reversibile del valore.

Con questo sistema, i punti di codifica che possono essere rappresentati vanno teoricamente da #x0 a #x7FFFFFFF (in particolare, secondo Unicode si arriva solo fino a #x10FFFF), salvo alcuni valori che sono stati esclusi espressamente. I punti di codifica esclusi più importanti sono #xFFFE e #xFFFF.

Le forme codificate del carattere che utilizzano le unità di codifica più piccole, richiedono l'uso di sequenze multiple di tali unità con maggiore frequenza. Per esempio, si può osservare il caso di UTF-8, in cui l'unità di codifica è il byte (un ottetto): mano a mano che il valore del punto di codifica cresce, è necessario utilizzare più unità di codifica per la sua rappresentazione.

È necessario sottolineare il fatto che i valori che compongono l'insieme dei punti di codifica, non vengono trasferiti tali e quali nella forma codificata, dal momento che ci possono essere delle limitazioni nella rappresentazione.

Allo stato attuale dello sviluppo dell'insieme di caratteri universale, le varie forme codificate del carattere possono utilizzare gli spazi seguenti:

UTF-8	da uno a sei unità di codifica da 8 bit (mentre secondo Unicode, che è più restrittivo dello standard ISO 10646, si hanno al massimo quattro unità);
UTF-16	da uno a due unità di codifica da 16 bit;
UTF-32	attualmente si prevede una sola unità di codifica da 32 bit.

47.7.1 UTF-8

UTF sta per *Unicode transformation format* e significa implicitamente che si tratta di una mappa di trasformazione da punti di codifica Unicode a unità di codifica (è già stato descritto il fatto che il numero che segue la sigla UTF-***n*** indica la dimensione in bit dell'unità di codifica).

In particolare, vale la pena di osservare un po' meglio UTF-8, essendo questo il cardine della transizione verso l'insieme di caratteri universale nei sistemi operativi in cui non è conveniente l'utilizzo di unità di codifica più grandi. In effetti, UTF-8 è un sistema molto complesso per rappresentare simboli di qualunque lingua diversa dall'inglese, perché richiede spesso l'utilizzo di più unità per un solo simbolo.

Le caratteristiche di UTF-8 sono le seguenti:

- i punti di codifica da #x0 a #x7F, corrispondenti in pratica all'ASCII, sono tradotti semplicemente in byte da 00_{16} a $7F_{16}$, esattamente come si fa già con l'ASCII stesso;
- i punti di codifica che vanno da #x80 in su, vengono tradotti in sequenze multiple di byte, ognuno dei quali ha il bit più significativo a uno, così da evitare che i byte da 00_{16} a $7F_{16}$ possano apparire all'interno delle sequenze multiple;
- il primo byte di una sequenza multipla che rappresenta un punto di codifica che vada da #x80 in su, contiene sempre valori nell'intervallo da $C0_{16}$ a FD_{16} e serve a indicare quanti byte vengono utilizzati per rappresentare il carattere;
- i byte di una sequenza multipla che sono successivi al primo contengono valori che vanno da 80_{16} a BF_{16};
- si possono definire sequenze di byte in numero massimo di sei;
- i valori FE_{16} e FF_{16} non sono mai usati.

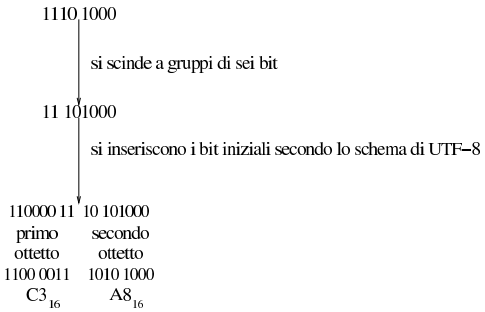
La tabella 47.34 dovrebbe chiarire meglio il concetto, abbinando i valori dei punti di codifica Unicode alle sequenze di byte con cui possono essere rappresentati. Si osservi che la lettera ***x*** serve a indicare un bit variabile.

Tabella 47.34. Sequenze multi-byte teoriche nell'UTF-8.

da	a	sequenze di ottetti
#x0	#x7F	0*xxxxxxx*
#x80	#x7FF	110*xxxxx* 10*xxxxxx*
#x800	#xFFFF	1110*xxxx* 10*xxxxxx* 10*xxxxxx*
#x10000	#x1FFFFF	11110*xxx* 10*xxxxxx* 10*xxxxxx* 10*xxxxxx*
#x200000	#x3FFFFFF	111110*xx* 10*xxxxxx* 10*xxxxxx* 10*xxxxxx* 10*xxxxxx*
#x4000000	#x7FFFFFFF	1111110*x* 10*xxxxxx* 10*xxxxxx* 10*xxxxxx* 10*xxxxxx* 10*xxxxxx*

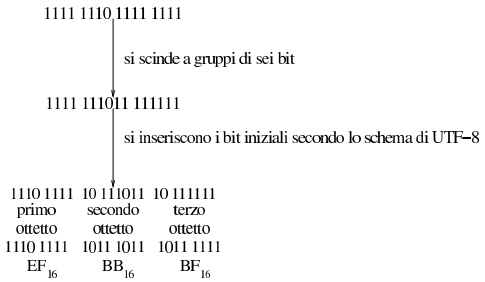
Un esempio dovrebbe chiarire ancora meglio il meccanismo. La lettera accentata «è» si rappresenta attraverso il punto di codifica #xE8, il quale in pratica si può rendere in binario come $1110\ 1000_2$, si traduce in UTF-8 come si vede nella figura 47.35.

Figura 47.35. #xE8 in UTF-8.



Un altro esempio interessante è il punto di codifica #xFEFF ($1111\ 1110\ 1111\ 1111_2$); lo si vede nella figura 47.36.

Figura 47.36. #xFEFF in UTF-8.



Da questo si dovrebbe intendere il passaggio a un numero superiore di byte.

In base al modello di UTF-8, si potrebbero realizzare anche sequenze più lunghe del necessario per rappresentare un punto di codifica. Evidentemente, è compito del software che le genera evitare di sprecare dello spazio inutilmente.

47.7.2 Schema di codifica e firma di riconoscimento

Di fronte a diverse forme codificate del carattere UTF, c'è la necessità di poterle identificare facilmente. Per questo si utilizza una sorta di firma iniziale, costituita in pratica dal punto di codifica #xFEFF, il quale, nella trasformazione in base allo schema di codifica del carattere, permette anche di controllare se l'ordine dei byte è normale o è stato invertito.

Il punto di codifica #xFEFF viene anche identificato con il nome ZWNBSP, ovvero *Zero width no-break space*; tuttavia, anche se si intende che si tratta di qualcosa di «innocuo» (uno spazio non interrompibile di ampiezza nulla), se è stato inserito come firma iniziale, non va inteso come parte del testo. Ciò significa che i programmi per la gestione di file di testo devono tenere conto che la firma iniziale va tolta prima di fare qualunque elaborazione (si pensi al concatenamento con un comando '`cat`' o simile).

Gli schemi di codifica del carattere riferiti alle forme codificate UTF, si possono precisare aggiungendo delle sigle alla fine del nome UTF-***n***. La tabella 47.37 mostra gli schemi di codifica UTF-***n****, assieme alla firma iniziale (quando questa è prevista).

Tabella 47.37. Schemi di codifica UTF-***n***.

Schema	Firma iniziale	Note
UTF-8	$EFBBBF_{16}$	In condizioni normali è prevista la firma iniziale.
UTF-8N		Si indica esplicitamente l'assenza della firma.
UTF-16		UTF-16 *big-endian* in modo predefinito.
UTF-16	$FEFF_{16}$	UTF-16 *big-endian*.
UTF-16	$FFFE_{16}$	UTF-16 *little-endian*.
UTF-16BE		UTF-16 *big-endian* senza firma.
UTF-16LE		UTF-16 *little-endian* senza firma.
UTF-32		UTF-32 *big-endian* in modo predefinito.
UTF-32	$0000FEFF_{16}$	UTF-32 *big-endian*.
UTF-32	$FFFE0000_{16}$	UTF-32 *little-endian*.
UTF-32BE		UTF-32 *big-endian* senza firma.
UTF-32LE		UTF-32 *little-endian* senza firma.

47.7.3 Tipi di dati nuovi

Si è già accennato al modo in cui un linguaggio di programmazione può gestire i punti di codifica di questo tipo. Tuttavia, non si può dimenticare il passato; così, in tutte le situazioni in cui il «carattere» è implicitamente un intero senza segno a 8 bit, è necessario usare un'altra definizione per i punti di codifica: il ***carattere esteso***, ovvero *wide char*. Nello stesso modo, dovendo parlare di stringhe, se c'è bisogno di chiarire che si tratta di una stringa secondo Unicode o ISO 10646, si parla di ***stringa estesa***, ovvero di *wide string*.

47.7.4 Apparenza e realtà

La disponibilità di un sistema di codifica che faccia riferimento a un repertorio simbolico molto ampio, risolve tanti problemi del passato in cui era necessario risparmiare. Per esempio, nell'ASCII tradizionale, il trattino è unico (non si distingue la sua lunghezza) ed è anche un segno «meno». Disponendo di un repertorio molto grande, diventa importante utilizzare il simbolo giusto in base al contesto. Per esempio, la lettera latina «A» maiuscola, è diversa dalla lettera greca alfa maiuscola, anche se i due simboli possono avere lo stesso aspetto.

La descrizione che viene abbinata ai punti di codifica serve proprio per questo, in modo da evitare confusione.

Per fare un esempio più convincente, si pensi alla lettera «ß» nell'insieme ISO 8859-1. Il nome abbinato a questa lettera è «LATIN SMALL LETTER SHARP S»; come si legge non si tratta della lettera greca beta, ma di qualcosa di diverso. Per la precisione è un legato che si usa nella lingua tedesca; in mancanza del segno tipografico può essere reso come «ss» (infatti si tratta di una lettera minuscola). Utilizzare questo simbolo al posto della lettera beta sarebbe un errore; infatti, un sistema di composizione o di lettura, potrebbe anche decidere di convertire il segno nella forma semplificata che è appena stata mostrata.

47.7.5 ASCII (ISO 646)

L'ASCII è una codifica molto semplice, in cui ogni punto di codifica corrisponde direttamente a un gruppo di 7 bit, inteso come un intero senza segno, senza bisogno di trasformazioni. Sulla base di questa codifica si sono sviluppate molte varianti, soprattutto a 8 bit. Tuttavia, oggi, quando si parla di ASCII si tende a fare riferimento prevalentemente allo standard originale, in cui si utilizzavano valori compresi tra 0 e 127, per rappresentare i quali bastano solo 7 bit. Eventualmente, volendo essere precisi, per fare riferimento all'ASCII tradizionale si può utilizzare la denominazione «US-ASCII».

L'ASCII non si occupa solo di definire la codifica dei segni tipografici, ma include anche dei codici di controllo, ai quali abbina un nome, ma senza potervi attribuire un significato univoco valido in tutti i contesti. Si tratta dei punti di codifica da 0 a 31 e del 127 in decimale (il punto di codifica 32 rappresenta lo spazio normale).

La tabella 47.38 mostra nel dettaglio la codifica ASCII.

Tabella 47.38. US-ASCII (ISO 646). L'ultima colonna si riferisce alla rappresentazione corrispondente nel linguaggio C.

Binario	Esadecimale	Ottale	Decimale	Carattere	<^***x***>	Linguaggio C
00000000_2	00_{16}	000_8	000_{10}	*<NUL>*		\0
00000001_2	01_{16}	001_8	001_{10}	*<SOH>*	*<^a>*	
00000010_2	02_{16}	002_8	002_{10}	*<STX>*	*<^b>*	
00000011_2	03_{16}	003_8	003_{10}	*<ETX>*	*<^c>*	
00000100_2	04_{16}	004_8	004_{10}	*<EOT>*	*<^d>*	
00000101_2	05_{16}	005_8	005_{10}	*<ENQ>*	*<^e>*	
00000110_2	06_{16}	006_8	006_{10}	*<ACK>*	*<^f>*	
00000111_2	07_{16}	007_8	007_{10}	*<BEL>*	*<^g>*	\a
00001000_2	08_{16}	010_8	008_{10}	*<BS>*	*<^h>*	\b
00001001_2	09_{16}	011_8	009_{10}	*<HT>*	*<^i>*	\t
00001010_2	$0A_{16}$	012_8	010_{10}	*<LF>*	*<^j>*	\n
00001011_2	$0B_{16}$	013_8	011_{10}	*<VT>*	*<^k>*	\v
00001100_2	$0C_{16}$	014_8	012_{10}	*<FF>*	*<^l>*	\f
00001101_2	$0D_{16}$	015_8	013_{10}	*<CR>*	*<^m>*	\r
00001110_2	$0E_{16}$	016_8	014_{10}	*<SO>*	*<^n>*	
00001111_2	$0F_{16}$	017_8	015_{10}	*<SI>*	*<^o>*	
00010000_2	10_{16}	020_8	016_{10}	*<DLE>*	*<^p>*	
00010001_2	11_{16}	021_8	017_{10}	*<DC1>*	*<^q>*	
00010010_2	12_{16}	022_8	018_{10}	*<DC2>*	*<^r>*	
00010011_2	13_{16}	023_8	019_{10}	*<DC3>*	*<^s>*	
00010100_2	14_{16}	024_8	020_{10}	*<DC4>*	*<^t>*	
00010101_2	15_{16}	025_8	021_{10}	*<NAK>*	*<^u>*	
00010110_2	16_{16}	026_8	022_{10}	*<SYN>*	*<^v>*	
00010111_2	17_{16}	027_8	023_{10}	*<ETB>*	*<^w>*	
00011000_2	18_{16}	030_8	024_{10}	*<CAN>*	*<^x>*	
00011001_2	19_{16}	031_8	025_{10}	*<EM>*	*<^y>*	
00011010_2	$1A_{16}$	032_8	026_{10}	*<SUB>*	*<^z>*	
00011011_2	$1B_{16}$	033_8	027_{10}	*<ESC>*	*<^[>*	
00011100_2	$1C_{16}$	034_8	028_{10}	*<FS>*	*<^\>*	

Binario	Esadecimale	Ottale	Decimale	Carattere	<^x>	Linguaggio C
00011101_2	$1D_{16}$	035_8	029_{10}	*<GS>*	*<^]>*	
00011110_2	$1E_{16}$	036_8	030_{10}	*<RS>*	*<^^>*	
00011111_2	$1F_{16}$	037_8	031_{10}	*<US>*	*<^_>*	
00100000_2	20_{16}	040_8	032_{10}	*<SP>*		
00100001_2	21_{16}	041_8	033_{10}	!		
00100010_2	22_{16}	042_8	034_{10}	"		
00100011_2	23_{16}	043_8	035_{10}	#		
00100100_2	24_{16}	044_8	036_{10}	$		
00100101_2	25_{16}	045_8	037_{10}	%		
00100110_2	26_{16}	046_8	038_{10}	&		
00100111_2	27_{16}	047_8	039_{10}	’		
00101000_2	28_{16}	050_8	040_{10}	(		
00101001_2	29_{16}	051_8	041_{10}	)		
00101010_2	$2A_{16}$	052_8	042_{10}	*		
00101011_2	$2B_{16}$	053_8	043_{10}	+		
00101100_2	$2C_{16}$	054_8	044_{10}	,		
00101101_2	$2D_{16}$	055_8	045_{10}	-		
00101110_2	$2E_{16}$	056_8	046_{10}	·		
00101111_2	$2F_{16}$	057_8	047_{10}	/		
00110000_2	30_{16}	060_8	048_{10}	0		
00110001_2	31_{16}	061_8	049_{10}	1		
00110010_2	32_{16}	062_8	050_{10}	2		
00110011_2	33_{16}	063_8	051_{10}	3		
00110100_2	34_{16}	064_8	052_{10}	4		
00110101_2	35_{16}	065_8	053_{10}	5		
00110110_2	36_{16}	066_8	054_{10}	6		
00110111_2	37_{16}	067_8	055_{10}	7		
00111000_2	38_{16}	070_8	056_{10}	8		
00111001_2	39_{16}	071_8	057_{10}	9		
00111010_2	$3A_{16}$	072_8	058_{10}	:		
00111011_2	$3B_{16}$	073_8	059_{10}	;		
00111100_2	$3C_{16}$	074_8	060_{10}	<		
00111101_2	$3D_{16}$	075_8	061_{10}	=		
00111110_2	$3E_{16}$	076_8	062_{10}	>		
00111111_2	$3F_{16}$	077_8	063_{10}	?		
01000000_2	40_{16}	100_8	064_{10}	@		
01000001_2	41_{16}	101_8	065_{10}	A		
01000010_2	42_{16}	102_8	066_{10}	B		
01000011_2	43_{16}	103_8	067_{10}	C		
01000100_2	44_{16}	104_8	068_{10}	D		
01000101_2	45_{16}	105_8	069_{10}	E		
01000110_2	46_{16}	106_8	070_{10}	F		
01000111_2	47_{16}	107_8	071_{10}	G		
01001000_2	48_{16}	110_8	072_{10}	H		
01001001_2	49_{16}	111_8	073_{10}	I		
01001010_2	$4A_{16}$	112_8	074_{10}	J		
01001011_2	$4B_{16}$	113_8	075_{10}	K		
01001100_2	$4C_{16}$	114_8	076_{10}	L		
01001101_2	$4D_{16}$	115_8	077_{10}	M		
01001110_2	$4E_{16}$	116_8	078_{10}	N		
01001111_2	$4F_{16}$	117_8	079_{10}	O		
01010000_2	50_{16}	120_8	080_{10}	P		
01010001_2	51_{16}	121_8	081_{10}	S		
01010010_2	52_{16}	122_8	082_{10}	R		
01010011_2	53_{16}	123_8	083_{10}	S		
01010100_2	54_{16}	124_8	084_{10}	T		
01010101_2	55_{16}	125_8	085_{10}	U		

Binario	Esadecimale	Ottale	Decimale	Carattere	<^x>	Linguaggio C
01010110_2	56_{16}	126_8	086_{10}	V		
01010111_2	57_{16}	127_8	087_{10}	W		
01011000_2	58_{16}	130_8	088_{10}	X		
01011001_2	59_{16}	131_8	089_{10}	Y		
01011010_2	$5A_{16}$	132_8	090_{10}	Z		
01011011_2	$5B_{16}$	133_8	091_{10}	[		
01011100_2	$5C_{16}$	134_8	092_{10}	\		\\
01011101_2	$5D_{16}$	135_8	093_{10}	]		
01011110_2	$5E_{16}$	136_8	094_{10}	^		
01011111_2	$5F_{16}$	137_8	095_{10}	–		
01100000_2	60_{16}	140_8	096_{10}	‘		
01100001_2	61_{16}	141_8	097_{10}	a		
01100010_2	62_{16}	142_8	098_{10}	b		
01100011_2	63_{16}	143_8	099_{10}	c		
01100100_2	64_{16}	144_8	100_{10}	d		
01100101_2	65_{16}	145_8	101_{10}	e		
01100110_2	66_{16}	146_8	102_{10}	f		
01100111_2	67_{16}	147_8	103_{10}	g		
01101000_2	68_{16}	150_8	104_{10}	h		
01101001_2	69_{16}	151_8	105_{10}	i		
01101010_2	$6A_{16}$	152_8	106_{10}	j		
01101011_2	$6B_{16}$	153_8	107_{10}	k		
01101100_2	$6C_{16}$	154_8	108_{10}	l		
01101101_2	$6D_{16}$	155_8	109_{10}	m		
01101110_2	$6E_{16}$	156_8	110_{10}	n		
01101111_2	$6F_{16}$	157_8	111_{10}	o		
01110000_2	70_{16}	160_8	112_{10}	p		
01110001_2	71_{16}	161_8	113_{10}	q		
01110010_2	72_{16}	162_8	114_{10}	r		
01110011_2	73_{16}	163_8	115_{10}	s		
01110100_2	74_{16}	164_8	116_{10}	t		
01110101_2	75_{16}	165_8	117_{10}	u		
01110110_2	76_{16}	166_8	118_{10}	v		
01110111_2	77_{16}	167_8	119_{10}	w		
01111000_2	78_{16}	170_8	120_{10}	x		
01111001_2	79_{16}	171_8	121_{10}	y		
01111010_2	$7A_{16}$	172_8	122_{10}	z		
01111011_2	$7B_{16}$	173_8	123_{10}	{		
01111100_2	$7C_{16}$	174_8	124_{10}	|		
01111101_2	$7D_{16}$	175_8	125_{10}	}		
01111110_2	$7E_{16}$	176_8	126_{10}	~		
01111111_2	$7F_{16}$	177_8	127_{10}	*<DEL>*		

47.7.6 ISO 8859-*n*

Le codifiche ISO 8859-***n***, dove ***n*** è un numero da 1 a 15, rappresentano per il passato l’evoluzione più coerente dell’ASCII, in quanto utilizzano tutte gli stessi punti di codifica iniziali da 0 a 127, corrispondenti esattamente all’ASCII originale.

Come nel caso dell’ASCII, non c’è distinzione tra punto di codifica e forma codificata del carattere; in questa situazione si usano valori fino a 255, attraverso un byte intero.

Le codifiche ISO 8859-***n*** introducono altri codici di controllo, nell’intervallo di punti di codifica che va da 128 a 159.

Per quanto riguarda le lingue occidentali, la codifica ISO 8859 più comune è ISO 8859-1, conosciuta anche come ISO Latin 1, rimpiazzata poi da ISO 8859-15 (ISO Latin 9), in cui si inserisce il simbolo dell’Euro al posto del simbolo di valuta generico.

La tabella 47.39 mostra nel dettaglio la codifica ISO 8859-1. Si tenga presente che nel caso di ISO 8859-15, il punto di codifica 164 viene abbinato al simbolo dell'Euro.

Tabella 47.39. ISO 8859-1.

Ottale	Decimale	Esadecimale	Carattere	Denominazione in inglese
240_8	160_{10}	$A0_{16}$		NO-BREAK SPACE
241_8	161_{10}	$A1_{16}$	¡	INVERTED EXCLAMATION MARK
242_8	162_{10}	$A2_{16}$	¢	CENT SIGN
243_8	163_{10}	$A3_{16}$	£	POUND SIGN
244_8	164_{10}	$A4_{16}$	□	CURRENCY SIGN
245_8	165_{10}	$A5_{16}$	¥	YEN SIGN
246_8	166_{10}	$A6_{16}$	¦	BROKEN BAR
247_8	167_{10}	$A7_{16}$	§	SECTION SIGN
250_8	168_{10}	$A8_{16}$	¨	DIAERESIS
251_8	169_{10}	$A9_{16}$	©	COPYRIGHT SIGN
252_8	170_{10}	AA_{16}	ª	FEMININE ORDINAL INDICATOR
253_8	171_{10}	AB_{16}	«	LEFT-POINTING DOUBLE ANGLE QUOTATION MARK
254_8	172_{10}	AC_{16}	¬	NOT SIGN
255_8	173_{10}	AD_{16}		SOFT HYPHEN
256_8	174_{10}	AE_{16}	®	REGISTERED SIGN
257_8	175_{10}	AF_{16}	¯	MACRON
260_8	176_{10}	$B0_{16}$	°	DEGREE SIGN
261_8	177_{10}	$B1_{16}$	±	PLUS-MINUS SIGN
262_8	178_{10}	$B2_{16}$	²	SUPERSCRIPT TWO
263_8	179_{10}	$B3_{16}$	³	SUPERSCRIPT THREE
264_8	180_{10}	$B4_{16}$	´	ACUTE ACCENT
265_8	181_{10}	$B5_{16}$	μ	MICRO SIGN
266_8	182_{10}	$B6_{16}$	¶	PILCROW SIGN
267_8	183_{10}	$B7_{16}$	·	MIDDLE DOT
270_8	184_{10}	$B8_{16}$	¸	CEDILLA
271_8	185_{10}	$B9_{16}$	¹	SUPERSCRIPT ONE
272_8	186_{10}	BA_{16}	º	MASCULINE ORDINAL INDICATOR
273_8	187_{10}	BB_{16}	»	RIGHT-POINTING DOUBLE ANGLE QUOTATION MARK
274_8	188_{10}	BC_{16}	¼	VULGAR FRACTION ONE QUARTER
275_8	189_{10}	BD_{16}	½	VULGAR FRACTION ONE HALF
276_8	190_{10}	BE_{16}	¾	VULGAR FRACTION THREE QUARTERS
277_8	191_{10}	BF_{16}	¿	INVERTED QUESTION MARK
300_8	192_{10}	$C0_{16}$	À	LATIN CAPITAL LETTER A WITH GRAVE
301_8	193_{10}	$C1_{16}$	Á	LATIN CAPITAL LETTER A WITH ACUTE
302_8	194_{10}	$C2_{16}$	Â	LATIN CAPITAL LETTER A WITH CIRCUMFLEX
303_8	195_{10}	$C3_{16}$	Ã	LATIN CAPITAL LETTER A WITH TILDE
304_8	196_{10}	$C4_{16}$	Ä	LATIN CAPITAL LETTER A WITH DIAERESIS
305_8	197_{10}	$C5_{16}$	Å	LATIN CAPITAL LETTER A WITH RING ABOVE
306_8	198_{10}	$C6_{16}$	Æ	LATIN CAPITAL LETTER AE
307_8	199_{10}	$C7_{16}$	Ç	LATIN CAPITAL LETTER C WITH CEDILLA
310_8	200_{10}	$C8_{16}$	È	LATIN CAPITAL LETTER E WITH GRAVE
311_8	201_{10}	$C9_{16}$	É	LATIN CAPITAL LETTER E WITH ACUTE
312_8	202_{10}	CA_{16}	Ê	LATIN CAPITAL LETTER E WITH CIRCUMFLEX
313_8	203_{10}	CB_{16}	Ë	LATIN CAPITAL LETTER E WITH DIAERESIS
314_8	204_{10}	CC_{16}	Ì	LATIN CAPITAL LETTER I WITH GRAVE
315_8	205_{10}	CD_{16}	Í	LATIN CAPITAL LETTER I WITH ACUTE
316_8	206_{10}	CE_{16}	Î	LATIN CAPITAL LETTER I WITH CIRCUMFLEX
317_8	207_{10}	CF_{16}	Ï	LATIN CAPITAL LETTER I WITH DIAERESIS
320_8	208_{10}	$D0_{16}$	Ð	LATIN CAPITAL LETTER ETH
321_8	209_{10}	$D1_{16}$	Ñ	LATIN CAPITAL LETTER N WITH TILDE
322_8	210_{10}	$D2_{16}$	Ò	LATIN CAPITAL LETTER O WITH GRAVE
323_8	211_{10}	$D3_{16}$	Ó	LATIN CAPITAL LETTER O WITH ACUTE
324_8	212_{10}	$D4_{16}$	Ô	LATIN CAPITAL LETTER O WITH CIRCUMFLEX
325_8	213_{10}	$D5_{16}$	Õ	LATIN CAPITAL LETTER O WITH TILDE
326_8	214_{10}	$D6_{16}$	Ö	LATIN CAPITAL LETTER O WITH DIAERESIS
327_8	215_{10}	$D7_{16}$	×	MULTIPLICATION SIGN
330_8	216_{10}	$D8_{16}$	Ø	LATIN CAPITAL LETTER O WITH STROKE
331_8	217_{10}	$D9_{16}$	Ù	LATIN CAPITAL LETTER U WITH GRAVE
332_8	218_{10}	DA_{16}	Ú	LATIN CAPITAL LETTER U WITH ACUTE
333_8	219_{10}	DB_{16}	Û	LATIN CAPITAL LETTER U WITH CIRCUMFLEX
334_8	220_{10}	DC_{16}	Ü	LATIN CAPITAL LETTER U WITH DIAERESIS
335_8	221_{10}	DD_{16}	Ý	LATIN CAPITAL LETTER Y WITH ACUTE
336_8	222_{10}	DE_{16}	Þ	LATIN CAPITAL LETTER THORN
337_8	223_{10}	DF_{16}	ß	LATIN SMALL LETTER SHARP S
340_8	224_{10}	$E0_{16}$	à	LATIN SMALL LETTER A WITH GRAVE
341_8	225_{10}	$E1_{16}$	á	LATIN SMALL LETTER A WITH ACUTE
342_8	226_{10}	$E2_{16}$	â	LATIN SMALL LETTER A WITH CIRCUMFLEX
343_8	227_{10}	$E3_{16}$	ã	LATIN SMALL LETTER A WITH TILDE
344_8	228_{10}	$E4_{16}$	ä	LATIN SMALL LETTER A WITH DIAERESIS
345_8	229_{10}	$E5_{16}$	å	LATIN SMALL LETTER A WITH RING ABOVE
346_8	230_{10}	$E6_{16}$	æ	LATIN SMALL LETTER AE
347_8	231_{10}	$E7_{16}$	ç	LATIN SMALL LETTER C WITH CEDILLA
350_8	232_{10}	$E8_{16}$	è	LATIN SMALL LETTER E WITH GRAVE
351_8	233_{10}	$E9_{16}$	é	LATIN SMALL LETTER E WITH ACUTE
352_8	234_{10}	EA_{16}	ê	LATIN SMALL LETTER E WITH CIRCUMFLEX
353_8	235_{10}	EB_{16}	ë	LATIN SMALL LETTER E WITH DIAERESIS
354_8	236_{10}	EC_{16}	ì	LATIN SMALL LETTER I WITH GRAVE
355_8	237_{10}	ED_{16}	í	LATIN SMALL LETTER I WITH ACUTE
356_8	238_{10}	EE_{16}	î	LATIN SMALL LETTER I WITH CIRCUMFLEX
357_8	239_{10}	EF_{16}	ï	LATIN SMALL LETTER I WITH DIAERESIS
360_8	240_{10}	$F0_{16}$	ð	LATIN SMALL LETTER ETH
361_8	241_{10}	$F1_{16}$	ñ	LATIN SMALL LETTER N WITH TILDE
362_8	242_{10}	$F2_{16}$	ò	LATIN SMALL LETTER O WITH GRAVE
363_8	243_{10}	$F3_{16}$	ó	LATIN SMALL LETTER O WITH ACUTE
364_8	244_{10}	$F4_{16}$	ô	LATIN SMALL LETTER O WITH CIRCUMFLEX
365_8	245_{10}	$F5_{16}$	õ	LATIN SMALL LETTER O WITH TILDE
366_8	246_{10}	$F6_{16}$	ö	LATIN SMALL LETTER O WITH DIAERESIS
367_8	247_{10}	$F7_{16}$	÷	DIVISION SIGN
370_8	248_{10}	$F8_{16}$	ø	LATIN SMALL LETTER O WITH STROKE
371_8	249_{10}	$F9_{16}$	ù	LATIN SMALL LETTER U WITH GRAVE
372_8	250_{10}	FA_{16}	ú	LATIN SMALL LETTER U WITH ACUTE
373_8	251_{10}	FB_{16}	û	LATIN SMALL LETTER U WITH CIRCUMFLEX
374_8	252_{10}	FC_{16}	ü	LATIN SMALL LETTER U WITH DIAERESIS
375_8	253_{10}	FD_{16}	ý	LATIN SMALL LETTER Y WITH ACUTE
376_8	254_{10}	FE_{16}	þ	LATIN SMALL LETTER THORN
377_8	255_{10}	FF_{16}	ÿ	LATIN SMALL LETTER Y WITH DIAERESIS

47.7.7 IBM Code Page *nnn* Character Set

I primi elaboratori IBM con sistema operativo Dos hanno utilizzato delle codifiche particolari, denominate *code page*. Due tra le varie codifiche sono molto importanti: CP 437 e CP 850. La prima era la codifica predefinita, mentre la seconda era quella più utile per le lingue europee occidentali.

Tabella 47.40. Prima parte comune delle codifiche CP 437 e CP 850.

Ottale	Decimale	Esadecimale	Codice corrispondente nell'insieme di caratteri universale	Aspetto
000_8	0_{10}	00_{16}	U+0000	*<NUL>*
001_8	1_{10}	01_{16}	U+263A	☐
002_8	2_{10}	02_{16}	U+263B	☐
003_8	3_{10}	03_{16}	U+2665	♥
004_8	4_{10}	04_{16}	U+2666	♦
005_8	5_{10}	05_{16}	U+2663	♣
006_8	6_{10}	06_{16}	U+2660	♠
007_8	7_{10}	07_{16}	U+2022	•
010_8	8_{10}	08_{16}	U+25D8	☐
011_8	9_{10}	09_{16}	U+25CB	☐
012_8	10_{10}	$0A_{16}$	U+25D9	☐
013_8	11_{10}	$0B_{16}$	U+2642	☐
014_8	12_{10}	$0C_{16}$	U+2640	☐
015_8	13_{10}	$0D_{16}$	U+266A	☐
016_8	14_{10}	$0E_{16}$	U+266B	☐
017_8	15_{10}	$0F_{16}$	U+263C	☐
020_8	16_{10}	10_{16}	U+25B6	☐
021_8	17_{10}	11_{16}	U+25C0	☐
022_8	18_{10}	12_{16}	U+2195	↕
023_8	19_{10}	13_{16}	U+203C	!!
024_8	20_{10}	14_{16}	U+00B6	¶
025_8	21_{10}	15_{16}	U+00A7	§
026_8	22_{10}	16_{16}	U+25AC	☐
027_8	23_{10}	17_{16}	U+21A8	☐
030_8	24_{10}	18_{16}	U+2191	↑
031_8	25_{10}	19_{16}	U+2193	v
032_8	26_{10}	$1A_{16}$	U+2192	→
033_8	27_{10}	$1B_{16}$	U+2190	←
034_8	28_{10}	$1C_{16}$	U+221F	☐
035_8	29_{10}	$1D_{16}$	U+2194	↔
036_8	30_{10}	$1E_{16}$	U+25B2	▲
037_8	31_{10}	$1F_{16}$	U+25BC	▼
040_8	32_{10}	20_{16}	U+0020	
041_8	33_{10}	21_{16}	U+0021	!
042_8	34_{10}	22_{16}	U+0022	"
043_8	35_{10}	23_{16}	U+0023	#
044_8	36_{10}	24_{16}	U+0024	$
045_8	37_{10}	25_{16}	U+0025	%
046_8	38_{10}	26_{16}	U+0026	&
047_8	39_{10}	27_{16}	U+0027	’
050_8	40_{10}	28_{16}	U+0028	(
051_8	41_{10}	29_{16}	U+0029	)
052_8	42_{10}	$2A_{16}$	U+002A	*
053_8	43_{10}	$2B_{16}$	U+002B	+
054_8	44_{10}	$2C_{16}$	U+002C	,
055_8	45_{10}	$2D_{16}$	U+002D	-
056_8	46_{10}	$2E_{16}$	U+002E	.
057_8	47_{10}	$2F_{16}$	U+002F	/
060_8	48_{10}	30_{16}	U+0030	0
061_8	49_{10}	31_{16}	U+0031	1
062_8	50_{10}	32_{16}	U+0032	2
063_8	51_{10}	33_{16}	U+0033	3
064_8	52_{10}	34_{16}	U+0034	4
065_8	53_{10}	35_{16}	U+0035	5
066_8	54_{10}	36_{16}	U+0036	6
067_8	55_{10}	37_{16}	U+0037	7
070_8	56_{10}	38_{16}	U+0038	8
071_8	57_{10}	39_{16}	U+0039	9
072_8	58_{10}	$3A_{16}$	U+003A	:
073_8	59_{10}	$3B_{16}$	U+003B	;
074_8	60_{10}	$3C_{16}$	U+003C	<
075_8	61_{10}	$3D_{16}$	U+003D	=
076_8	62_{10}	$3E_{16}$	U+003E	>
077_8	63_{10}	$3F_{16}$	U+003F	?
100_8	64_{10}	40_{16}	U+0040	@
101_8	65_{10}	41_{16}	U+0041	A
102_8	66_{10}	42_{16}	U+0042	B
103_8	67_{10}	43_{16}	U+0043	C
104_8	68_{10}	44_{16}	U+0044	D
105_8	69_{10}	45_{16}	U+0045	E
106_8	70_{10}	46_{16}	U+0046	F
107_8	71_{10}	47_{16}	U+0047	G
110_8	72_{10}	48_{16}	U+0048	H
111_8	73_{10}	49_{16}	U+0049	I
112_8	74_{10}	$4A_{16}$	U+004A	J
113_8	75_{10}	$4B_{16}$	U+004B	K
114_8	76_{10}	$4C_{16}$	U+004C	L
115_8	77_{10}	$4D_{16}$	U+004D	M
116_8	78_{10}	$4E_{16}$	U+004E	N
117_8	79_{10}	$4F_{16}$	U+004F	O
120_8	80_{10}	50_{16}	U+0050	P
121_8	81_{10}	51_{16}	U+0051	Q
122_8	82_{10}	52_{16}	U+0052	R
123_8	83_{10}	53_{16}	U+0053	S
124_8	84_{10}	54_{16}	U+0054	T
125_8	85_{10}	55_{16}	U+0055	U
126_8	86_{10}	56_{16}	U+0056	V
127_8	87_{10}	57_{16}	U+0057	W
130_8	88_{10}	58_{16}	U+0058	X
131_8	89_{10}	59_{16}	U+0059	Y
132_8	90_{10}	$5A_{16}$	U+005A	Z
133_8	91_{10}	$5B_{16}$	U+005B	[
134_8	92_{10}	$5C_{16}$	U+005C	\
135_8	93_{10}	$5D_{16}$	U+005D	]
136_8	94_{10}	$5E_{16}$	U+005E	^
137_8	95_{10}	$5F_{16}$	U+005F	–
140_8	96_{10}	60_{16}	U+0060	‘
141_8	97_{10}	61_{16}	U+0061	a
142_8	98_{10}	62_{16}	U+0062	b
143_8	99_{10}	63_{16}	U+0063	c
144_8	100_{10}	64_{16}	U+0064	d
145_8	101_{10}	65_{16}	U+0065	e
146_8	102_{10}	66_{16}	U+0066	f
147_8	103_{10}	67_{16}	U+0067	g
150_8	104_{10}	68_{16}	U+0068	h
151_8	105_{10}	69_{16}	U+0069	i
152_8	106_{10}	$6A_{16}$	U+006A	j
153_8	107_{10}	$6B_{16}$	U+006B	k
154_8	108_{10}	$6C_{16}$	U+006C	l
155_8	109_{10}	$6D_{16}$	U+006D	m
156_8	110_{10}	$6E_{16}$	U+006E	n
157_8	111_{10}	$6F_{16}$	U+006F	o
160_8	112_{10}	70_{16}	U+0070	p
161_8	113_{10}	71_{16}	U+0071	q
162_8	114_{10}	72_{16}	U+0072	r
163_8	115_{10}	73_{16}	U+0073	s
164_8	116_{10}	74_{16}	U+0074	t
165_8	117_{10}	75_{16}	U+0075	u
166_8	118_{10}	76_{16}	U+0076	v
167_8	119_{10}	77_{16}	U+0077	w
170_8	120_{10}	78_{16}	U+0078	x
171_8	121_{10}	79_{16}	U+0079	y
172_8	122_{10}	$7A_{16}$	U+007A	z
173_8	123_{10}	$7B_{16}$	U+007B	{
174_8	124_{10}	$7C_{16}$	U+007C	|
175_8	125_{10}	$7D_{16}$	U+007D	}
176_8	126_{10}	$7E_{16}$	U+007E	~
177_8	127_{10}	$7F_{16}$	U+2302	☐

Tabella 47.41. Seconda parte della codifica CP 437.

Ottale	Decimale	Esadecimale	Codice corrispondente nell'insieme di caratteri universale	Aspetto
200_8	128_{10}	80_{16}	U+00C7	Ç
201_8	129_{10}	81_{16}	U+00FC	ü
202_8	130_{10}	82_{16}	U+00E9	é
203_8	131_{10}	83_{16}	U+00E2	â
204_8	132_{10}	84_{16}	U+00E4	ä
205_8	133_{10}	85_{16}	U+00E3	ã

Ottale	Decimale	Esadecimale	Codice corrispondente nell'insieme di caratteri universale	Aspetto
206_8	134_{10}	86_{16}	U+00E5	å
207_8	135_{10}	87_{16}	U+00E7	ç
210_8	136_{10}	88_{16}	U+00EA	ê
211_8	137_{10}	89_{16}	U+00EB	ë
212_8	138_{10}	$8A_{16}$	U+00E8	è
213_8	139_{10}	$8B_{16}$	U+00EF	ï
214_8	140_{10}	$8C_{16}$	U+00EE	î
215_8	141_{10}	$8D_{16}$	U+00EC	ì
216_8	142_{10}	$8E_{16}$	U+00C4	Ä
217_8	143_{10}	$8F_{16}$	U+00C5	Å
220_8	144_{10}	90_{16}	U+00C9	É
221_8	145_{10}	91_{16}	U+00E6	æ
222_8	146_{10}	92_{16}	U+00C6	Æ
223_8	147_{10}	93_{16}	U+00F4	ô
224_8	148_{10}	94_{16}	U+00F6	ö
225_8	149_{10}	95_{16}	U+00F2	ò
226_8	150_{10}	96_{16}	U+00FB	û
227_8	151_{10}	97_{16}	U+00F9	ù
230_8	152_{10}	98_{16}	U+00FF	ÿ
231_8	153_{10}	99_{16}	U+00D6	Ö
232_8	154_{10}	$9A_{16}$	U+00DC	Ü
233_8	155_{10}	$9B_{16}$	U+00F8	ø
234_8	156_{10}	$9C_{16}$	U+00A3	£
235_8	157_{10}	$9D_{16}$	U+00A5	¥
236_8	158_{10}	$9E_{16}$	U+20A7	□
237_8	159_{10}	$9F_{16}$	U+0192	*f*
240_8	160_{10}	$A0_{16}$	U+00E1	á
241_8	161_{10}	$A1_{16}$	U+00ED	í
242_8	162_{10}	$A2_{16}$	U+00F3	ó
243_8	163_{10}	$A3_{16}$	U+00FA	ú
244_8	164_{10}	$A4_{16}$	U+00F1	ñ
245_8	165_{10}	$A5_{16}$	U+00D1	Ñ
246_8	166_{10}	$A6_{16}$	U+00AA	ª
247_8	167_{10}	$A7_{16}$	U+00BA	º
250_8	168_{10}	$A8_{16}$	U+00BF	¿
251_8	169_{10}	$A9_{16}$	U+00AE	®
252_8	170_{10}	AA_{16}	U+00AC	¬
253_8	171_{10}	AB_{16}	U+00BD	½
254_8	172_{10}	AC_{16}	U+00BC	¼
255_8	173_{10}	AD_{16}	U+00A1	¡
256_8	174_{10}	AE_{16}	U+00AB	«
257_8	175_{10}	AF_{16}	U+00BB	»
260_8	176_{10}	$B0_{16}$	U+2591	□
261_8	177_{10}	$B1_{16}$	U+2592	□
262_8	178_{10}	$B2_{16}$	U+2593	□
263_8	179_{10}	$B3_{16}$	U+2502	|
264_8	180_{10}	$B4_{16}$	U+2524	|
265_8	181_{10}	$B5_{16}$	U+2561	□
266_8	182_{10}	$B6_{16}$	U+2562	□
267_8	183_{10}	$B7_{16}$	U+2556	□
270_8	184_{10}	$B8_{16}$	U+2555	□
271_8	185_{10}	$B9_{16}$	U+2563	□
272_8	186_{10}	BA_{16}	U+2551	□
273_8	187_{10}	BB_{16}	U+2557	□
274_8	188_{10}	BC_{16}	U+255D	□
275_8	189_{10}	BD_{16}	U+255C	□
276_8	190_{10}	BE_{16}	U+255B	□
277_8	191_{10}	BF_{16}	U+2510	·
300_8	192_{10}	$C0_{16}$	U+2514	‘
301_8	193_{10}	$C1_{16}$	U+2534	□
302_8	194_{10}	$C2_{16}$	U+252C	□
303_8	195_{10}	$C3_{16}$	U+251C	|
304_8	196_{10}	$C4_{16}$	U+2500	-
305_8	197_{10}	$C5_{16}$	U+253C	□
306_8	198_{10}	$C6_{16}$	U+255E	□
307_8	199_{10}	$C7_{16}$	U+255F	□
310_8	200_{10}	$C8_{16}$	U+255A	□
311_8	201_{10}	$C9_{16}$	U+2554	□
312_8	202_{10}	CA_{16}	U+2569	□
313_8	203_{10}	CB_{16}	U+2566	□

Ottale	Decimale	Esadecimale	Codice corrispondente nell'insieme di caratteri universale	Aspetto
314_8	204_{10}	CC_{16}	U+2560	□
315_8	205_{10}	CD_{16}	U+2550	□
316_8	206_{10}	CE_{16}	U+256C	□
317_8	207_{10}	CF_{16}	U+2567	□
320_8	208_{10}	$D0_{16}$	U+2568	□
321_8	209_{10}	$D1_{16}$	U+2564	□
322_8	210_{10}	$D2_{16}$	U+2565	□
323_8	211_{10}	$D3_{16}$	U+2559	□
324_8	212_{10}	$D4_{16}$	U+2558	□
325_8	213_{10}	$D5_{16}$	U+2552	□
326_8	214_{10}	$D6_{16}$	U+2553	□
327_8	215_{10}	$D7_{16}$	U+256B	□
330_8	216_{10}	$D8_{16}$	U+256A	□
331_8	217_{10}	$D9_{16}$	U+2518	’
332_8	218_{10}	DA_{16}	U+250C	·
333_8	219_{10}	DB_{16}	U+2588	□
334_8	220_{10}	DC_{16}	U+2584	□
335_8	221_{10}	DD_{16}	U+258C	□
336_8	222_{10}	DE_{16}	U+2590	□
337_8	223_{10}	DF_{16}	U+2580	□
340_8	224_{10}	$E0_{16}$	U+03B1	α
341_8	225_{10}	$E1_{16}$	U+00DF	ß
342_8	226_{10}	$E2_{16}$	U+0393	Γ
343_8	227_{10}	$E3_{16}$	U+03C0	π
344_8	228_{10}	$E4_{16}$	U+03A3	Σ
345_8	229_{10}	$E5_{16}$	U+03C3	σ
346_8	230_{10}	$E6_{16}$	U+00B5	*μ*
347_8	231_{10}	$E7_{16}$	U+03C4	τ
350_8	232_{10}	$E8_{16}$	U+03A6	Φ
351_8	233_{10}	$E9_{16}$	U+0398	Θ
352_8	234_{10}	EA_{16}	U+03A9	Ω
353_8	235_{10}	EB_{16}	U+03B4	δ
354_8	236_{10}	EC_{16}	U+221E	∞
355_8	237_{10}	ED_{16}	U+03C6	ϕ
356_8	238_{10}	EE_{16}	U+03B5	ε
357_8	239_{10}	EF_{16}	U+2229	∩
360_8	240_{10}	$F0_{16}$	U+2261	≡
361_8	241_{10}	$F1_{16}$	U+00B1	±
362_8	242_{10}	$F2_{16}$	U+2265	≥
363_8	243_{10}	$F3_{16}$	U+2264	≤
364_8	244_{10}	$F4_{16}$	U+2320	⌠
365_8	245_{10}	$F5_{16}$	U+2321	⌡
366_8	246_{10}	$F6_{16}$	U+00F7	÷
367_8	247_{10}	$F7_{16}$	U+2248	≈
370_8	248_{10}	$F8_{16}$	U+00B0	°
371_8	249_{10}	$F9_{16}$	U+2219	∙
372_8	250_{10}	FA_{16}	U+00B7	·
373_8	251_{10}	FB_{16}	U+221A	√
374_8	252_{10}	FC_{16}	U+207F	^{n}
375_8	253_{10}	FD_{16}	U+00B2	^{2}
376_8	254_{10}	FE_{16}	U+25A0	■
377_8	255_{10}	FF_{16}	U+00A0	

Tabella 47.42. Seconda parte della codifica CP 850.

Ottale	Decimale	Esadecimale	Codice corrispondente nell'insieme di caratteri universale	Aspetto
200_8	128_{10}	80_{16}	U+00C7	Ç
201_8	129_{10}	81_{16}	U+00FC	ü
202_8	130_{10}	82_{16}	U+00E9	é
203_8	131_{10}	83_{16}	U+00E2	â
204_8	132_{10}	84_{16}	U+00E4	ä
205_8	133_{10}	85_{16}	U+00E3	ã
206_8	134_{10}	86_{16}	U+00E5	å
207_8	135_{10}	87_{16}	U+00E7	ç
210_8	136_{10}	88_{16}	U+00EA	ê
211_8	137_{10}	89_{16}	U+00EB	ë
212_8	138_{10}	$8A_{16}$	U+00E8	è
213_8	139_{10}	$8B_{16}$	U+00EF	ï

Ottale	Decimale	Esadecimale	Codice corrispondente nell'insieme di caratteri universale	Aspetto
214_8	140_{10}	$8C_{16}$	U+00EE	î
215_8	141_{10}	$8D_{16}$	U+00EC	ì
216_8	142_{10}	$8E_{16}$	U+00C4	Ä
217_8	143_{10}	$8F_{16}$	U+00C5	Å
220_8	144_{10}	90_{16}	U+00C9	É
221_8	145_{10}	91_{16}	U+00E6	æ
222_8	146_{10}	92_{16}	U+00C6	Æ
223_8	147_{10}	93_{16}	U+00F4	ô
224_8	148_{10}	94_{16}	U+00F6	ö
225_8	149_{10}	95_{16}	U+00F2	ò
226_8	150_{10}	96_{16}	U+00FB	û
227_8	151_{10}	97_{16}	U+00F9	ù
230_8	152_{10}	98_{16}	U+00FF	ÿ
231_8	153_{10}	99_{16}	U+00D6	Ö
232_8	154_{10}	$9A_{16}$	U+00DC	Ü
233_8	155_{10}	$9B_{16}$	U+00F8	ø
234_8	156_{10}	$9C_{16}$	U+00A3	£
235_8	157_{10}	$9D_{16}$	U+00D8	Ø
236_8	158_{10}	$9E_{16}$	U+00D7	×
237_8	159_{10}	$9F_{16}$	U+0192	*f*
240_8	160_{10}	$A0_{16}$	U+00E1	á
241_8	161_{10}	$A1_{16}$	U+00ED	í
242_8	162_{10}	$A2_{16}$	U+00F3	ó
243_8	163_{10}	$A3_{16}$	U+00FA	ú
244_8	164_{10}	$A4_{16}$	U+00F1	ñ
245_8	165_{10}	$A5_{16}$	U+00D1	Ñ
246_8	166_{10}	$A6_{16}$	U+00AA	ª
247_8	167_{10}	$A7_{16}$	U+00BA	º
250_8	168_{10}	$A8_{16}$	U+00BF	¿
251_8	169_{10}	$A9_{16}$	U+00AE	®
252_8	170_{10}	AA_{16}	U+00AC	¬
253_8	171_{10}	AB_{16}	U+00BD	½
254_8	172_{10}	AC_{16}	U+00BC	¼
255_8	173_{10}	AD_{16}	U+00A1	¡
256_8	174_{10}	AE_{16}	U+00AB	«
257_8	175_{10}	AF_{16}	U+00BB	»
260_8	176_{10}	$B0_{16}$	U+2591	□
261_8	177_{10}	$B1_{16}$	U+2592	□
262_8	178_{10}	$B2_{16}$	U+2593	□
263_8	179_{10}	$B3_{16}$	U+2502	|
264_8	180_{10}	$B4_{16}$	U+2524	|
265_8	181_{10}	$B5_{16}$	U+00C1	Á
266_8	182_{10}	$B6_{16}$	U+00C1	Á
267_8	183_{10}	$B7_{16}$	U+00C0	À
270_8	184_{10}	$B8_{16}$	U+00A9	©
271_8	185_{10}	$B9_{16}$	U+2563	□
272_8	186_{10}	BA_{16}	U+2551	□
273_8	187_{10}	BB_{16}	U+2557	□
274_8	188_{10}	BC_{16}	U+255D	□
275_8	189_{10}	BD_{16}	U+00A2	¢
276_8	190_{10}	BE_{16}	U+00A5	¥
277_8	191_{10}	BF_{16}	U+2510	·
300_8	192_{10}	$C0_{16}$	U+2514	‘
301_8	193_{10}	$C1_{16}$	U+2534	□
302_8	194_{10}	$C2_{16}$	U+252C	□
303_8	195_{10}	$C3_{16}$	U+251C	|
304_8	196_{10}	$C4_{16}$	U+2500	-
305_8	197_{10}	$C5_{16}$	U+253C	□
306_8	198_{10}	$C6_{16}$	U+00E3	ã
307_8	199_{10}	$C7_{16}$	U+00C3	Ã
310_8	200_{10}	$C8_{16}$	U+255A	□
311_8	201_{10}	$C9_{16}$	U+2554	□
312_8	202_{10}	CA_{16}	U+2569	□
313_8	203_{10}	CB_{16}	U+2566	□
314_8	204_{10}	CC_{16}	U+2560	□
315_8	205_{10}	CD_{16}	U+2550	□
316_8	206_{10}	CE_{16}	U+256C	□
317_8	207_{10}	CF_{16}	U+00A4	□
320_8	208_{10}	$D0_{16}$	U+00F0	ð
321_8	209_{10}	$D1_{16}$	U+00D0	Ð

Ottale	Decimale	Esadecimale	Codice corrispondente nell'insieme di caratteri universale	Aspetto
322_8	210_{10}	$D2_{16}$	U+00CA	Ê
323_8	211_{10}	$D3_{16}$	U+00CB	Ë
324_8	212_{10}	$D4_{16}$	U+00C8	È
325_8	213_{10}	$D5_{16}$	U+0131	ı
326_8	214_{10}	$D6_{16}$	U+00CD	Í
327_8	215_{10}	$D7_{16}$	U+00CE	Î
330_8	216_{10}	$D8_{16}$	U+00CF	Ï
331_8	217_{10}	$D9_{16}$	U+2518	’
332_8	218_{10}	DA_{16}	U+250C	.
333_8	219_{10}	DB_{16}	U+2588	□
334_8	220_{10}	DC_{16}	U+2584	□
335_8	221_{10}	DD_{16}	U+00A6	¦
336_8	222_{10}	DE_{16}	U+00CC	Ì
337_8	223_{10}	DF_{16}	U+2580	□
340_8	224_{10}	$E0_{16}$	U+00D3	Ó
341_8	225_{10}	$E1_{16}$	U+00DF	ß
342_8	226_{10}	$E2_{16}$	U+00D4	Ô
343_8	227_{10}	$E3_{16}$	U+00D2	Ò
344_8	228_{10}	$E4_{16}$	U+00F5	õ
345_8	229_{10}	$E5_{16}$	U+00D5	Õ
346_8	230_{10}	$E6_{16}$	U+00B5	*µ*
347_8	231_{10}	$E7_{16}$	U+00FE	þ
350_8	232_{10}	$E8_{16}$	U+00DE	Þ
351_8	233_{10}	$E9_{16}$	U+00DA	Ú
352_8	234_{10}	EA_{16}	U+00DB	Û
353_8	235_{10}	EB_{16}	U+00D9	Ù
354_8	236_{10}	EC_{16}	U+00FD	ý
355_8	237_{10}	ED_{16}	U+00DD	Ý
356_8	238_{10}	EE_{16}	U+00AF	¯
357_8	239_{10}	EF_{16}	U+00B4	´
360_8	240_{10}	$F0_{16}$	U+00AD	
361_8	241_{10}	$F1_{16}$	U+00B1	±
362_8	242_{10}	$F2_{16}$	U+2017	□
363_8	243_{10}	$F3_{16}$	U+00BE	¾
364_8	244_{10}	$F4_{16}$	U+00B6	¶
365_8	245_{10}	$F5_{16}$	U+00A7	§
366_8	246_{10}	$F6_{16}$	U+00F7	÷
367_8	247_{10}	$F7_{16}$	U+00B8	¸
370_8	248_{10}	$F8_{16}$	U+00B0	°
371_8	249_{10}	$F9_{16}$	U+00A8	¨
372_8	250_{10}	FA_{16}	U+00B7	·
373_8	251_{10}	FB_{16}	U+00B9	1
374_8	252_{10}	FC_{16}	U+00B3	3
375_8	253_{10}	FD_{16}	U+00B2	2
376_8	254_{10}	FE_{16}	U+25A0	■
377_8	255_{10}	FF_{16}	U+00A0	

47.7.8 Utilizzo di «ascii»

Il programma ‘**ascii**’[15] consente di ottenere facilmente informazioni su un certo carattere appartenente alla codifica ASCII, ovvero ai primi 127 punti di codifica.

```
ascii [opzioni] [carattere]...
```

Generalmente, si utilizza il programma specificando nella riga di comando uno o più caratteri, di cui si vogliono maggiori informazioni. I caratteri possono essere indicati letteralmente (ammesso che ciò sia possibile), oppure attraverso una forma alternativa, purché sia stata prevista.

Tabella 47.43. Alcune modalità con cui si può indicare un carattere nella riga di comando.

Forma	Descrizione
x	Se possibile, il carattere può essere indicato in forma letterale.
^*x*	La maggior parte dei caratteri di controllo può essere specificata secondo la forma ‘**^***x*’.

Forma	Descrizione
`\`***x***	Alcuni caratteri di controllo possono essere rappresentati in questo modo, che si rifà alle convenzioni del linguaggio C.
nome	I caratteri di controllo possono essere indicati per nome.
`0x`***n*** `x`***n*** `\x`***n***	Si tratta di forme equivalenti per indicare il punto di codifica del carattere in esadecimale.
`0d`***n*** `d`***n*** `\d`***n***	Si tratta di forme equivalenti per indicare il punto di codifica del carattere in decimale.
`0o`***n*** `o`***n*** `\o`***n***	Si tratta di forme equivalenti per indicare il punto di codifica del carattere in ottale.
`0b`***n*** `b`***n*** `\b`***n***	Si tratta di forme equivalenti per indicare il punto di codifica del carattere in binario.

Le opzioni della riga di comando non vengono descritte; si veda eventualmente la pagina di manuale *ascii(1)*. Vengono mostrati alcuni esempi.

`$` **`ascii`** [*Invio*]

In questo modo si ottiene una guida rapida all'utilizzo del programma, assieme al riepilogo della tabella ASCII. Si osservi che si ottiene lo stesso risultato con l'opzione '**`-h`**'.

```
Usage: ascii [-dxohv] [-t] [char-alias...]
   -t = one-line output  -d = Decimal table  -o = octal table
   -x = hex table -h = This help screen
   -v = version information
Prints all aliases of an ASCII character. Args may be chars,
C \-escapes, English names, ^-escapes, ASCII mnemonics, or
numerics in decimal/octal/hex.

Dec Hex    Dec Hex    Dec Hex  Dec Hex   Dec Hex   Dec Hex
  0 00 NUL  22 16 SYN  44 2C ,  66 42 B   88 58 X  110 6E n
  1 01 SOH  23 17 ETB  45 2D -  67 43 C   89 59 Y  111 6F o
  2 02 STX  24 18 CAN  46 2E .  68 44 D   90 5A Z  112 70 p
  3 03 ETX  25 19 EM   47 2F /  69 45 E   91 5B [  113 71 q
  4 04 EOT  26 1A SUB  48 30 0  70 46 F   92 5C \  114 72 r
  5 05 ENQ  27 1B ESC  49 31 1  71 47 G   93 5D ]  115 73 s
  6 06 ACK  28 1C FS   50 32 2  72 48 H   94 5E ^  116 74 t
  7 07 BEL  29 1D GS   51 33 3  73 49 I   95 5F _  117 75 u
  8 08 BS   30 1E RS   52 34 4  74 4A J   96 60 `  118 76 v
  9 09 HT   31 1F US   53 35 5  75 4B K   97 61 a  119 77 w
 10 0A LF   32 20      54 36 6  76 4C L   98 62 b  120 78 x
 11 0B VT   33 21 !    55 37 7  77 4D M   99 63 c  121 79 y
 12 0C FF   34 22 "    56 38 8  78 4E N  100 64 d  122 7A z
 13 0D CR   35 23 #    57 39 9  79 4F O  101 65 e  123 7B {
 14 0E SO   36 24 $    58 3A :  80 50 P  102 66 f  124 7C |
 15 0F SI   37 25 %    59 3B ;  81 51 Q  103 67 g  125 7D }
 16 10 DLE  38 26 &    60 3C <  82 52 R  104 68 h  126 7E ~
 17 11 DC1  39 27 '    61 3D =  83 53 S  105 69 i  127 7F DEL
 18 12 DC2  40 28 (    62 3E >  84 54 T  106 6A j
 19 13 DC3  41 29 )    63 3F ?  85 55 U  107 6B k
 20 14 DC4  42 2A *    64 40 @  86 56 V  108 6C l
 21 15 NAK  43 2B +    65 41 A  87 57 W  109 6D m
```

`$` **`ascii ETX`** [*Invio*]

L'argomento rappresenta il nome di un codice di controllo secondo le convenzioni della codifica ASCII. Si può osservare che questo carattere si può rappresentare, convenzionalmente, anche con la sequenza <^C>.

```
ASCII 0/3 is decimal 003, hex 03, octal 003, bits 00000011:
called ^C, ETX Official name: End of Text
```


`$` **`ascii ^c`** [*Invio*]

Si ottiene esattamente lo stesso risultato dell'esempio precedente.

`$` **`ascii a b c`** [*Invio*]

Mostra le informazioni sui tre caratteri inseriti:

```
ASCII 6/1 is decimal 097, hex 61, octal 141, bits 01100001:
prints as 'a'
Official name: Miniscule a
Other names: Small a, Lowercase a

ASCII 6/2 is decimal 098, hex 62, octal 142, bits 01100010:
prints as 'b'
Official name: Miniscule b
Other names: Small b, Lowercase b

ASCII 6/3 is decimal 099, hex 63, octal 143, bits 01100011:
prints as 'c'
Official name: Miniscule c
Other names: Small c, Lowercase c
```

`$` **`ascii -t a b c`** [*Invio*]

Mostra alcune informazioni sui caratteri indicati come argomento finale:

```
6/1   97   0x61   0o141   01100001
6/2   98   0x62   0o142   01100010
6/3   99   0x63   0o143   01100011
```

`$` **`ascii -s abc`** [*Invio*]

Mostra alcune informazioni sui caratteri che compongono la stringa fornita come argomento finale:

```
6/1   97   0x61   0o141   01100001
6/2   98   0x62   0o142   01100010
6/3   99   0x63   0o143   01100011
```

47.7.9 Utilizzo di «unicode»

Il programma '**`unicode`**'[16] consente di ottenere facilmente informazioni su un certo carattere, secondo la classificazione della codifica universale.

`unicode` [***opzioni***] [***carattere_o_stringa***]…

Gli argomenti finali della riga di comando possono essere caratteri singoli, oppure stringhe; l'indicazione di una stringa implica la richiesta di avere informazioni su tutti i caratteri che contiene. I caratteri cercati possono essere indicati letteralmente, oppure attraverso altre forme, come documentato nella pagina di manuale *unicode(1)*, ma generalmente conviene limitarsi all'uso della notazione tipica per Unicode: '**`U+`*****n***', dove ***n*** è un numero del punto di codifica espresso in esadecimale.

Tabella 47.49. Alcune modalità con cui si possono indicare i caratteri nella riga di comando.

Forma	Descrizione
x	Se possibile, il carattere può essere indicato in forma letterale.
`U+`***n*** ***n***	Rappresenta il punto di codifica in esadecimale.
`U+`***n***`..U+`***m*** ***n***`..`***m***	Rappresenta un intervallo di punti di codifica e si usa per ottenere la tabella dei codici relativi.

Sono disponibili delle opzioni, che generalmente non è necessario utilizzare; eventualmente si può consultare la pagina di manuale *unicode(1)*. Vengono mostrati alcuni esempi.

`$` **`unicode à`** [*Invio*]

Richiede informazioni sulla lettera «à» (a minuscola con accento grave).

```
U+00E0 LATIN SMALL LETTER A WITH GRAVE
UTF-8: c3 a0   UTF-16BE: 00e0   Decimal: &#224;
à (À)
Uppercase: U+00C0
Category: Ll (Letter, Lowercase)
Bidi: L (Left-to-Right)
Decomposition: 0061 0300
```

`$ unicode U+0416` [*Invio*]

Richiede informazioni sul punto di codifica U+0416:

```
U+0416 CYRILLIC CAPITAL LETTER ZHE
UTF-8: d0 96   UTF-16BE: 0416   Decimal: &#1046;
Ж (ж)
Lowercase: U+0436
Category: Lu (Letter, Uppercase)
Bidi: L (Left-to-Right)
```

`$ unicode 0416` [*Invio*]

Esattamente come nell'esempio precedente.

`$ unicode U+0005..U+0010` [*Invio*]

Richiede di visualizzare il blocco di codici che comprende i punti di codifica da U+0005 a U+0010; in pratica, visualizza i punti di codifica da U+0000 a U+00FF:

```
      .0 .1 .2 .3 .4 .5 .6 .7 .8 .9 .A .B .C .D .E .F
000.
001.
002.    !  "  #  $  %  &  ’  (  )  *  +  ,  -  .  /
003.  0  1  2  3  4  5  6  7  8  9  :  ;  <  =  >  ?
004.  @  A  B  C  D  E  F  G  H  I  J  K  L  M  N  O
005.  P  Q  R  S  T  U  V  W  X  Y  Z  [  \  ]  ^  _
006.  ‘  a  b  c  d  e  f  g  h  i  j  k  l  m  n  o
007.  p  q  r  s  t  u  v  w  x  y  z  {  |  }  ~
008.
009.
00A.    ¡  ¢  £  □  ¥  ¦  §  ¨  ©  ª  «  ¬     ®  ¯
00B.  °  ±  ²  ³  ´  µ  ¶  ·  ¸  ¹  º  »  ¼  ½  ¾  ¿
00C.  À  Á  Â  Ã  Ä  Å  Æ  Ç  È  É  Ê  Ë  Ì  Í  Î  Ï
00D.  Ð  Ñ  Ò  Ó  Ô  Õ  Ö  ×  Ø  Ù  Ú  Û  Ü  Ý  Þ  ß
00E.  à  á  â  ã  ä  å  æ  ç  è  é  ê  ë  ì  í  î  ï
00F.  ð  ñ  ò  ó  ô  õ  ö  ÷  ø  ù  ú  û  ü  ý  þ  ÿ
```

47.7.10 Gucharmap

Attraverso il sistema grafico X è possibile utilizzare Gucharmap,[17] con il quale si ottengono informazioni sui codici dei caratteri, secondo la codifica universale, in modo pratico.

```
gucharmap
```

Il programma '**gucharmap**' si avvia senza opzioni particolari e si interagisce con lui, attraverso un'interfaccia grafica.

Figura 47.53. Gucharmap con la selezione del punto di codifica U+0D24.

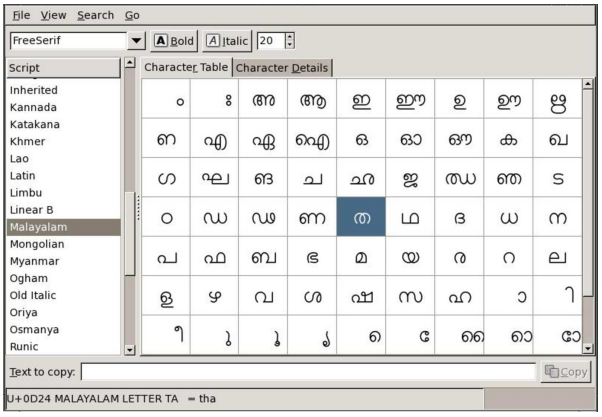
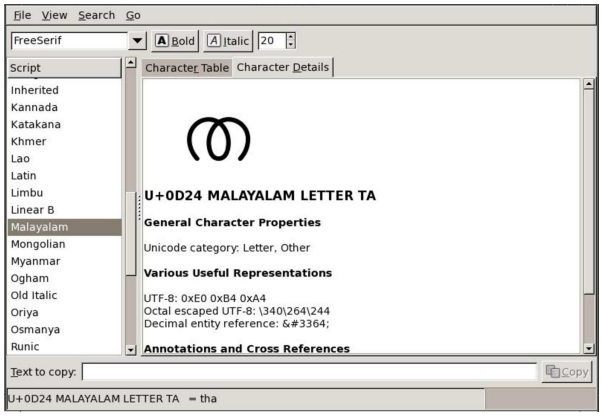


Figura 47.54. Gucharmap con le informazioni dettagliate del punto di codifica U+0D24.

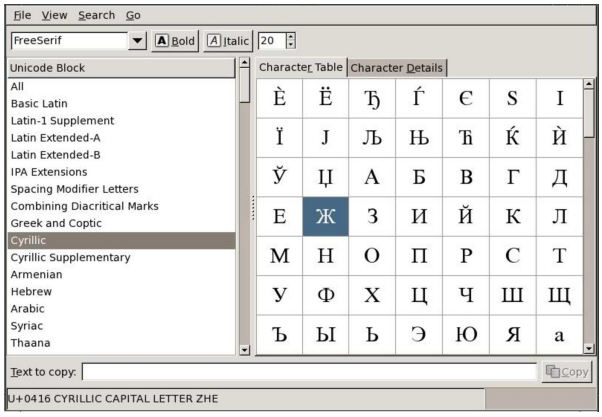


Una volta selezionato il gruppo di codici di proprio interesse, è facile copiare i caratteri desiderati trascinandoli nella riga di testo che appare nella parte inferiore della finestra, come si vede nella figura 47.55. Da lì è poi possibile copiarli nell'applicazione di proprio interesse, utilizzando i metodi consueti per le operazioni di «taglia-copia-incolla».

Figura 47.55. Copia di alcuni caratteri nella riga inferiore.

I caratteri della codifica universale possono essere classificati in base alla scrittura (*script*), oppure in base ai blocchi utilizzati nella codifica stessa. La figura 47.56 mostra una porzione del blocco cirillico (da U+0400 a U+04FF).

Figura 47.56. Classificazione dei codici secondo i blocchi della codifica universale.



47.8 Trasformazione della codifica

La presenza di codifiche diverse comporta spesso la necessità di trasformare i file di testo da una codifica a un'altra, oppure di far lavorare un programma in un contesto che utilizza una codifica diversa da quella impostata generalmente nel sistema.

47.8.1 Recode

Recode[18] è un programma per la conversione di file da un insieme di caratteri a un altro. Recode non si limita semplicemente a questo; spesso è in grado di intervenire su codifiche composte da sequenze di caratteri, anche se in queste situazioni il suo utilizzo si complica e i risultati non sono sempre garantiti.

```
recode [opzioni] codifica_prima . . codifica_dopo [file...]
```

Osservando lo schema sintattico mostrato, si può vedere che è necessario indicare il tipo di conversione attraverso due parole chiave: la prima serve a stabilire il modo in cui sono codificati i dati in ingresso, la seconda stabilisce in che modo li si vuole trasformare in uscita.

Uno o più file in ingresso possono essere indicati alla fine della riga di comando, facendo sì che Recode tenti di sovrascriverli; in alternativa, un file in ingresso può essere fornito attraverso lo standard input, ottenendo il risultato della conversione dallo standard output.

Bisogna essere prudenti con Recode quando si indicano i file nella riga di comando, perché la conversione potrebbe essere irreversibile.

Nel suo piccolo, Recode è un programma complesso. Questa sezione mostra solo alcuni aspetti banali, mentre per sfruttare bene tutte le sue potenzialità è necessario leggere la documentazione originale: *info recode*.

Tabella 47.57. Alcune opzioni.

Opzione	Descrizione
`-l` [*codifica*] `--list`[=*codifica*]	Questa opzione, usata senza argomento, mostra l'elenco delle codifiche gestite; con l'aggiunta del nome di una codifica, si ottiene invece una tabella della codifica stessa.
`-g` `--graphics`	Questa opzione riguarda la conversione dall'insieme di caratteri '**`IBM-PC`**', a un altro tipo, dove si vuole tentare di trasformare in qualche modo i simboli grafici tipici di quella codifica. È evidente che questa conversione è irreversibile.

Le codifiche da utilizzare nelle conversioni sono indicate attraverso la notazione ***codifica_prima**..**codifica_dopo***, come si vede nello schema sintattico introduttivo. Le parole chiave utilizzate per questo possono essere indicate indifferentemente utilizzando le lettere minuscole o maiuscole. L'elenco delle codifiche (e quindi delle trasformazioni possibili) è molto lungo e potrebbe essere ottenuto un riepilogo attraverso l'opzione '**`-l`**' da sola. Tuttavia, sarebbe meglio leggere prima ciò che è stato annotato nel documento *info recode* al riguardo, per non rischiare di trovarsi poi nei pasticci.

Tabella 47.58. Alcune parole chiave che si possono usare per individuare le codifiche.

Codifica	Descrizione

`IBM437`
`437`
`cp437`
	Rappresenta la codifica IBM usata normalmente nel Dos. Quando si converte da questa codifica a un'altra, i codici di interruzione di riga vengono lasciati inalterati.

`IBM-PC` `ibmpc`	È praticamente la stessa codifica IBM437, con la differenza che quando si converte da questa codifica a un'altra, i codici di interruzione di riga vengono trasformati.

`IBM850`
`850`
`cp850`
	Rappresenta la codifica IBM usata normalmente nel Dos per la localizzazione europea.

`ISO_8859-1:1987`
`ISO_8859-1`
`ISO-8859-1`
`CP819`
`IBM819`
`iso-ir-100`
`l1`
`latin1`
	Si riferisce alla codifica ISO 8859-1.

`UTF-8` `UTF8`	Si riferisce alla codifica UTF-8.

Segue la descrizione di alcuni esempi.

```
$ recode -a IBM437 [Invio]
```

Mostra tutte le possibilità di abbinamento con la codifica IBM437.

```
$ recode -a IBM-PC [Invio]
```

Mostra tutte le possibilità di abbinamento con la codifica IBM-PC.

```
$ recode IBM-PC..ISO_8859-1 lettera [Invio]
```

Converte il file '`lettera`' dalla codifica IBM-PC, con codice di interruzione di riga pari a *<CR><LF>* a ISO 8859-1, con codice di interruzione di riga pari a *<LF>* soltanto, sovrascrivendo il file.

```
$ recode IBM-PC..ISO_8859-1 < lettera > lettera2 [Invio]
```

Converte il file '`lettera`' dalla codifica ISO 8859-1, con codice di interruzione di riga pari a *<LF>*, a IBM-PC, con codice di interruzione di riga pari a *<CR><LF>*, generando il file '`lettera2`'.

```
$ recode -g IBM-PC..ISO_8859-1 < schema1 > schema2 [Invio]
```

Converte il file '`schema1`' dalla codifica IBM-PC a ISO 8859-1, generando il file '`schema2`', tentando di convertire anche i simboli grafici. Anche in questo caso ha luogo la conversione del codice di interruzione di riga.

47.8.2 Iconv

Iconv[19] è un programma previsto dallo standard dei sistemi Unix e serve a convertire un file da un insieme di caratteri a un altro. Lo schema sintattico seguente mostra l'utilizzo normale del programma:

```
iconv -f da_codifica -t a_codifica [altre_opzioni] [file...]
```

La codifica viene indicata attraverso una parola chiave e l'elenco completo degli insiemi di caratteri conosciuti si ottiene con l'opzione '**`-l`**', usata da sola:

```
iconv -l
```

Il file da convertire viene indicato normalmente alla fine della riga di comando, ma in sua mancanza viene usato lo standard input; generalmente il risultato della conversione viene emesso attraverso lo standard output, a meno che sia usata espressamente l'opzione '**`-o`**', con la quale si indica il nome di un file da creare per questo scopo.

Tabella 47.59. Alcune opzioni.

Opzione	Descrizione
`-f` ***codifica_di_partenza*** `--from-code` ***codifica_di_partenza***	Specifica la codifica di origine.
`-t` ***codifica_di_destinazione*** `--to-code` ***codifica_di_destinazione***	Specifica la codifica del file da generare.
`-l` `--list`	Elenca le parole chiave riconosciute che individuano le varie codifiche gestibili dal programma. Si usa questa opzione da sola.
`-o` ***file_da_generare*** `--output` ***file_da_generare***	Richiede di fornire il risultato nel file specificato, senza usare per questo lo standard output.

Le parole chiave utilizzate per individuare la codifica possono essere indicate indifferentemente utilizzando le lettere minuscole o maiuscole. Si deve osservare che non tutte le trasformazioni sono possibili; pertanto, di fronte a richieste impossibili, il programma si rifiuta di procedere, così come si rifiuta di farlo se il file in ingresso non è conforme alla codifica dichiarata per questo.

Tabella 47.60. Alcune parole chiave che si possono usare per individuare le codifiche.

Codifica	Descrizione
`IBM437` `437` `CP437`	Rappresenta la codifica IBM usata normalmente nel Dos.
`IBM850` `850` `CP850`	Rappresenta la codifica IBM usata normalmente nel Dos per la localizzazione europea.
`LATIN1` `8859_1` `ISO-8859-1` `ISO_8859-1` `ISO_8859-1:1987` `ISO8859-1` `ISO88591`	Si riferisce alla codifica ISO 8859-1.
`UTF-8` `UTF8`	Si riferisce alla codifica UTF-8.

Segue la descrizione di alcuni esempi.

`$` **`iconv --list`** [*Invio*]

Mostra i nomi di tutte le codifiche gestibili.

`$` **`iconv -f LATIN1 -t UTF-8 lettera > lettera2`** [*Invio*]

Converte il file '`lettera`', scritto con la codifica ISO 8859-1, a UTF-8, generando il file '`lettera2`'.

`$` **`cat lettera | iconv -f LATIN1 -t UTF-8 > lettera2`** [*Invio*]

Esattamente come nell'esempio precedente, con la differenza che il file '`lettera`' viene fornito attraverso lo standard input.

`$` **`iconv -f BIG5 -t UTF-8 lettera > lettera2`** [*Invio*]

Converte il file '`lettera`', scritto con la codifica BIG5 (usata per la lingua cinese), a UTF-8, generando il file '`lettera2`'.

47.8.3 Utilizzo di «luit»

Il programma '**`luit`**', il quale fa parte di X (capitolo 28), è un filtro che si utilizza per avviare un altro programma, quando il proprio terminale a caratteri è configurato in modo da gestire la codifica UTF-8 (sia per la tastiera, sia per lo schermo) e il programma in questione utilizza una codifica differente:

`luit` [***opzioni***] [`--`] [***programma*** [***argomenti***]]

Come si può intuire dal modello sintattico, in mancanza dell'indicazione di un programma da avviare, '**`luit`**' avvia una shell. Al posto di descrivere le opzioni di questo programma, vengono mostrati alcuni esempi, a cominciare da quello più semplice, in cui il controllo della conversione avviene semplicemente attraverso la configurazione della variabile ***LC_ALL***:

`$` **`LC_ALL=en_US.ISO-8859-1 luit`** ***mio_programma*** [*Invio*]

In questo caso si avvia il programma ***mio_programma*** specificando per lui la variabile di ambiente ***LC_ALL*** con il valore che si può vedere. Il programma '**`luit`**' fa in modo che i dati provenienti dalla tastiera siano convertiti da UTF-8 a ISO 8859-1, facendo l'opposto per i dati diretti dal programma allo schermo. Si osservi, comunque, che la configurazione locale del tipo '**`en_US.ISO-8859-1`**' deve essere disponibile nel sistema.

`$` **`LC_ALL=en_US luit -encoding "ISO 8859-1"`** ***mio_programma*** [*Invio*]

In questo caso, si rende esplicita la codifica con cui deve funzionare il programma '**`mio_programma`**', attraverso l'opzione '**`-encoding`**', secondo la notazione prevista da '**`luit`**'. Per la precisione, si può ottenere l'elenco di tutte le codifiche previste, secondo la notazione di '**`luit`**', con l'opzione '**`-list`**':

`$` **`luit -list`** [*Invio*]

Si osservi che quando ci si collega a un elaboratore remoto, nel quale non è prevista una configurazione locale con una codifica UTF-8, è necessario usare '**`luit`**' come già mostrato, per esempio così:

`$` **`LC_ALL=en_US luit -encoding "ISO 8859-1" ssh`** ***nodo*** [*Invio*]

Nell'esempio si può riconoscere l'uso del programma '**`ssh`**', ovvero di Secure Shell.

Si osservi che negli esempi è stata usata la variabile di ambiente ***LC_ALL***, perché questa prevale su tutte le variabili ***LC_**** e anche su ***LANG***.

47.9 Analisi lessicale

Gli errori che si possono fare scrivendo un testo sono di vario tipo, ma quelli puramente lessicali, ovvero ciò che potrebbe essere classificato come errore di battitura, rappresentano i meno importanti. Tuttavia, si tratta pur sempre di una buona percentuale nell'insieme globale di errori che può contenere un testo.

Un programma banale che sia in grado di mostrare le parole che risultano semplicemente sconosciute, è già un buon aiuto verso l'obiettivo dello scrivere in modo corretto.

Un programma di analisi lessicale è utile quando si può gestire un dizionario personale, perché non si possono escludere le eccezioni da un testo, come per esempio il nome o il cognome di una persona, un indirizzo o una sigla particolare. In presenza di documenti di grandi dimensioni, diventa necessario gestire un dizionario specifico per ognuno di questi, in modo da non interferire con l'analisi di altri in cui certi termini, ammissibili da una parte, non possono esistere dall'altra.

47.9.1 Ispell

Ispell[20] è un programma di scansione lessicale che permette la realizzazione di dizionari contenenti anche indicazioni sulle possibili aggregazioni di parole (si pensi alla lingua tedesca in cui le parole sono generate spesso dall'unione di altre).

Lo studio di questa caratteristica di Ispell riguarda chi vuole realizzare un dizionario standard per un linguaggio particolare: generico o specifico di un certo settore. Qui si intende mostrare un uso semplificato di questo programma, in cui si utilizzano dizionari standard e si generano i propri dizionari personali specifici per ciò che si fa.

47.9.1.1 Dizionari

Generalmente, il pacchetto di distribuzione di Ispell contiene un dizionario predefinito (di solito per la lingua inglese). Dovrebbe trattarsi del file '`/usr/lib/ispell/default.hash`' (che comunque di solito è un collegamento simbolico a un altro file). Nella stessa directory vanno collocati altri file per altre lingue, o per linguaggi specifici. Questi file, terminanti con l'estensione '`.hash`', sono ottenuti a partire da una coppia di file di testo, attraverso la compilazione con '**`buildhash`**', ogni volta che si cambia piattaforma.

Il dizionario italiano, in forma sorgente, si compone di due file sorgenti: '`italian.aff`' e '`italian.words`'. Il primo dei due contiene la tabella *affix*, la quale in pratica rappresenta una serie di regole sull'insieme dei caratteri ammissibili e sulla possibile unione di parti di parole, mentre il secondo è l'elenco di parole vero e proprio. Queste parole elencate, contengono a volte dei riferimenti aggiuntivi indicati dopo una barra obliqua ('`/`') che hanno valore in base alle definizioni della tabella *affix*. L'approfondimento sulla sintassi del file *affix* è utile solo se si vuole realizzare un dizionario hash specifico, mentre l'utilizzatore normale può ignorare questo problema. La compilazione dei file sorgenti in modo da ottenere un dizionario hash si ottiene con il comando seguente:

```
$ buildhash italian.words italian.aff italian.hash [ Invio ]
```

Si ottiene il file '`italian.hash`', da collocare nella directory '`/usr/lib/ispell/`'. Se si intende utilizzare sistematicamente questo dizionario, si può predisporre la variabile di ambiente ***DICTIONARY***, assegnandovi il nome del file: '`italian.hash`'. In alternativa, si può usare '**`ispell`**' con l'opzione '**`-d`**', come nell'esempio seguente (l'estensione '`.hash`' è predefinita e può essere omessa).

```
$ ispell -d italian documento.txt [ Invio ]
```

I dizionari personali sono invece una cosa diversa: si tratta di un elenco di termini, scritto con le stesse modalità di un sorgente, senza un file *affix* a fianco (o meglio, utilizzando quello del dizionario hash a cui si fa riferimento). Normalmente, tali file personali sono aggiornati da Ispell, quando questo viene usato in modo interattivo. Il nome predefinito del dizionario personale è '`~/.ispell_`***linguaggio***'. Per esempio, se si utilizza il dizionario standard predefinito, viene generato e utilizzato il file '`~/.ispell_default`' (nella directory personale), a meno di specificare un nome diverso con le opzioni.

Si osservi che in aggiunta ai file personali ci possono essere dei file più specifici, legati alla directory corrente: '`./.ispell_`***linguaggio***'.

47.9.1.2 Avvio e opzioni fondamentali

Il modello seguente rappresenta una semplificazione estrema della sintassi dell'eseguibile '**`ispell`**', però, prima di apprendere il funzionamento delle particolarità di questo programma, è meglio comprendere le sue possibilità fondamentali:

```
ispell [opzioni] file_da_analizzare
```


Ispell può funzionare in modo interattivo, oppure no. In teoria, è possibile anche realizzare un programma che sfrutti le funzionalità di Ispell attraverso un condotto; in pratica, si tratta in questo caso dell'utilizzo meno importante che si può fare di Ispell.

Opzione	Descrizione
`-d` ***dizionario_hash***	Permette di specificare un file dizionario differente da quello predefinito (che di solito è '`english.hash`'). Il nome del file viene indicato generalmente senza estensione e senza percorso, facendo implicitamente riferimento alla directory '`/usr/lib/ispell/`' e a file con estensione '`.hash`'.
`-p` ***dizionario_personale***	Permette di specificare un dizionario personale differente da quello predefinito (che di solito è '`~/.ispell_`…').
`-W` ***n_caratteri***	Specifica la lunghezza delle parole che non devono essere prese in considerazione. In pratica, da quel numero di caratteri in giù, si considerano tutte valide.
`-x`	Evita la creazione di una copia di sicurezza. Senza indicare questa opzione, dovrebbe essere salvata una copia del file originale aggiungendo al suo nome l'estensione '`.bak`'.
`-b`	Si tratta dell'opzione opposta a '**`-x`**', in quanto permette di forzare la richiesta di creazione di una copia di sicurezza.
`-t`	Fa in modo che il testo da analizzare sia considerato un sorgente TeX, o LaTeX, per il quale si devono ignorare i codici di composizione e possibilmente anche alcune indicazioni che sono solo funzionali a TeX, dal momento che non riguardano il contenuto del testo. Questa dovrebbe essere la modalità predefinita di funzionamento. In generale, questa modalità va bene anche per il testo puro e semplice, purché non ci siano barre oblique inverse che possano essere confuse con comandi di TeX.
`-n`	Fa in modo che il testo da analizzare sia considerato un sorgente Nroff o Troff, per il quale si devono ignorare i codici di composizione. La possibilità di distinguere i codici di composizione di TeX, *roff, o altro, dipende anche dal file *affix* del dizionario utilizzato.

47.9.1.3 Funzionamento interattivo

Il funzionamento normale di Ispell è interattivo. Generalmente viene fatta una copia di sicurezza del file analizzato, con un nome che termina con l'aggiunta dell'estensione '`.bak`', quindi Ispell permette di modificare il contenuto del file originale, in base alle scelte dell'utente.

Figura 47.62. Funzionamento interattivo di Ispell.

```
    stai              File: lettera

Ciao come stai?

00: stab     09: st-AI
01: stag
02: staid
03: stain
04: stair
05: Stan
06: star
07: stay
08: st AI

[SP] <number> R)epl A)ccept I)nsert L)ookup U)ncap Q)uit
e(X)it or ? for help
```

La figura 47.62 mostra il caso di un file, denominato '`lettera`', contenente una frase normalissima, in cui la parola «stai» non viene

riconosciuta. In effetti, si suppone di avere utilizzato il dizionario hash predefinito, ovvero quello inglese.

La parola '**stai**' viene evidenziata se le caratteristiche del terminale lo consentono; in ogni caso, viene indicata a parte, all'inizio (come si vede dall'esempio). Se possibile, Ispell elenca una serie di alternative possibili, in base alle affinità che può avere il termine sconosciuto con altre parole contenute nel dizionario. Questo elenco è numerato, in modo da permetterne la selezione. Nella parte bassa dello schermo appare un menù riepilogativo degli altri comandi a disposizione; comandi che si richiamano prevalentemente con la semplice pressione di tasti o combinazioni di tasti mnemonici.

Comando	Descrizione
[*Spazio*]	Fa in modo che Ispell accetti la parola temporaneamente. Se successivamente Ispell ne trova ancora, queste vengono segnalate di nuovo.
[*R*] [*r*]	Richiede la sostituzione della parola errata con un'altra che deve essere inserita subito dopo. Se anche la nuova parola non sembra valida, questa viene segnalata ugualmente da Ispell. La sostituzione riguarda solo quell'occorrenza particolare; se viene ritrovato ancora lo stesso errore, Ispell continua a segnalarlo.
[*A*] [*a*]	Fa sì che Ispell ignori la parola per tutto il resto del documento.
[*I*] [*i*]	Fa sì che Ispell accetti la parola e la inserisca nel dizionario personale, esattamente com'è, rispettando maiuscole e minuscole.
[*U*] [*u*]	Fa sì che Ispell accetti la parola e la inserisca nel dizionario personale, senza distinguere tra maiuscole e minuscole.
[*0*] [*1*] ... [*0*][*0*] [*0*][*1*] ...	La selezione di un numero fa riferimento alle voci proposte come parole alternative a quella errata. Con questa selezione di intende ottenere la sostituzione delle parole. È importante osservare che, se l'elenco supera le nove unità, la selezione avviene con due cifre numeriche. L'esempio che appare nella figura mostra questo caso: per indicare la parola '**stag**', occorre la sequenza [*0*][*1*].
[*X*] [*x*]	Conclude il lavoro completando la scrittura del file e ignorando altri errori eventuali. Chiude anche il file del dizionario personale, mantenendo le voci aggiunte fino a quel punto.
[*Q*] [*q*]	Termina immediatamente, lasciando inalterato il file, senza conservare i termini eventualmente annotati per l'aggiunta nel dizionario personale.
[*Ctrl l*] [*Ctrl L*]	Ripulisce lo schermo.

Per quanto riguarda il funzionamento interattivo di Ispell, sono importanti due opzioni.

Opzione	Descrizione
`-M`	Richiede espressamente la visualizzazione del menù riassuntivo dei comandi interattivi. Di solito, tale menù appare in modo predefinito, a meno di avere compilato Ispell con opzioni particolari.
`-N`	Fa in modo che il menù riepilogativo dei comandi non venga visualizzato.

Segue la descrizione di alcuni comandi.

- `$ ispell -d italian lettera` [*Invio*]

  Analizza il file '`lettera`' utilizzando il dizionario hash '**italian**', ovvero, il file '`/usr/lib/ispell/italian.hash`', e il dizionario personale predefinito: '`~/.ispell_italian`'.

- `$ ispell -d italian -p mio lettera` [*Invio*]

  Come nell'esempio precedente, ma in questo caso si utilizza il dizionario personale rappresentato dal file '`./mio`'.

47.9.1.4 Funzionamento non interattivo

Quando Ispell funziona in modo non interattivo, si limita a generare un elenco di termini, anche ripetuti, che risultano sconosciuti in base al dizionario. Ispell può anche essere utilizzato attraverso un altro programma, quando si indica l'opzione '**-a**', ma si tratta di un modo un po' complicato che qui non viene descritto.

Per ottenere l'elenco dei termini sconosciuti, si utilizza l'opzione '**-l**'. Per esempio, questa possibilità di Ispell può essere sfruttata per produrre rapidamente un dizionario personale.

Se si dispone di un testo della cui esattezza si è certi, si può ottenere da Ispell l'elenco dei termini da lui sconosciuti, generando poi un dizionario personale con tutte queste eccezioni. Si procede nel modo seguente:

```
$ ispell -d italian -l < romanzo > mio_dizionario [Invio]
```

In questo modo, tutti i termini contenuti nel file '`./romanzo`' che non risultano dal dizionario hash '**italian**', vengono emessi attraverso lo standard output e diretti nel file '`./mio_dizionario`'.

```
$ sort -f < mio_dizionario > dizionario1 [Invio]
```

In questo modo si riordina l'elenco di parole ottenuto, generando il file '`./dizionario1`', dove l'opzione '**-f**' serve a non distinguere tra lettere minuscole e maiuscole, anche se restano i doppioni. Con questo elenco si vuole generare un dizionario personale, eliminando questi doppioni ed eventualmente generando altre semplificazioni.

```
$ munchlist -s italian -l italian.aff dizionario1 ↩
↪> dizionario2 [Invio]
```

In questo modo, si ottiene il compattamento del file '`./dizionario1`', in base a quanto già contenuto del dizionario hash '**italian**' e secondo le regole del file *affix* '`./italian.aff`', generando il file '`./dizionario2`', che finalmente può essere utilizzato come dizionario personale.

In alternativa, si può anche tentare di dare in pasto a Ispell il file ottenuto dopo l'ordinamento, senza filtrarlo attraverso '**munchlist**'; lasciando che sia Ispell stesso a eliminare i doppioni.

47.9.1.5 Programmi di servizio di contorno a Ispell

Ispell si compone di diversi file binari. Il più importante è '**ispell**', come si è visto, ma altri sono necessari per la gestione dei file di dizionario. Si è già accennato a '**buildhash**' e a '**munchlist**', il cui utilizzo è il caso di riepilogare.

```
buildhash dizionario_sorgente file_affix dizionario_hash
```

```
munchlist [-l file_affix] [-s dizionario_hash] [elenco_da_ridurre] ↩
↪> elenco_ridotto
```

Quelle mostrate sono le sintassi semplificate di questi due programmi. Di più può essere appreso dalla lettura di *ispell(1)*. Segue la descrizione di alcuni esempi.

- `$ munchlist mio_dizionario > dizionario` [*Invio*]

  Utilizza il dizionario hash e il file *affix* standard per ridurre l'elenco contenuto nel file '`./mio_dizionario`', generando il file '`./dizionario`'.

- `$ munchlist -s italian -l ./italian.aff mio_dizionario ↩`
  `↪> dizionario` [*Invio*]

  Utilizza il dizionario hash '**italian**' ('`/usr/lib/ispell/italian.hash`') e il file *affix* '`./italian.aff`' per ridurre l'elenco contenuto nel file '`./mio_dizionario`', generando il file '`./dizionario`'.

- `$ buildhash italian.words italian.aff italian.hash` [*Invio*]

  Genera il dizionario hash '`./italian.hash`', a partire dall'elenco '`./italian.words`' e dal file *affix* '`./italian.aff`'.

47.9.1.6 Gestione dei dizionari personali

L'utilizzo occasionale di Ispell richiede la presenza di un dizionario hash e probabilmente di uno personale predefinito, che quasi sicuramente è '`~/.ispell_italian`'. Ma la correzione ortografica basata esclusivamente su un dizionario è tanto più efficace quanto minore è il numero delle parole previste, ovvero, quanto più specifico è il dizionario utilizzato.

Di fronte alla realizzazione di un documento di un certo impegno, o di una serie di documenti che trattano dello stesso genere di cose, potrebbe essere conveniente utilizzare un dizionario personale specifico per quel progetto, eventualmente partendo da un dizionario hash praticamente vuoto.[21]

Per realizzare un dizionario «vuoto», adatto a qualunque linguaggio che utilizzi la codifica ISO 8859-1, si potrebbe partire dal file *affix* che contiene solo le righe seguenti, il cui unico scopo è quello di ammettere l'uso di tutte le lettere accentate e speciali.[22]

```
# minimo.aff
# Accetta qualunque carattere accentato e speciale di
# ISO 8859-1.

wordchars       [a-z]   [A-Z]
wordchars       [à-\376]        [À-\336]
wordchars       [\337]
wordchars       [\377]

prefixes

suffixes
```

Le parole chiave '**`prefixes`**' e '**`suffixes`**' sono obbligatorie, ma il file è ancora incompleto (viene segnalato dai programmi come '**`buildhash`**' e '**`munchlist`**'), anche se funziona ugualmente per lo scopo che ci si prefigge qui.

Volendo esagerare, se le cifre numeriche possono avere un ruolo nella composizione delle parole che si vogliono controllare, si può aggiungere anche la riga seguente, tenendo conto che però poi '**`munchlist`**' non funziona tanto bene.[23]

```
wordchars       [0-9]
```

A fianco di questo si deve creare un elenco di parole che ne contenga almeno una, come nell'esempio seguente:

```
Linux
```

Si suppone che il file *affix* sia stato nominato '`minimo.aff`' e che l'elenco sia '`minimo.sml`'. Per creare il file hash, si procede come è già stato presentato più volte.

```
$ buildhash minimo.sml minimo.aff minimo.hash [Invio]
```

Pur con una segnalazione di errore, dovuta all'estrema semplicità del file *affix*, si ottiene il file '`minimo.hash`' nella directory corrente. Questo file hash può essere usato solo per testi normali, senza codici di composizione di alcun tipo, dal momento che il file *affix* mostrato non è stato predisposto per questo.

Se si dispone di un documento ritenuto sicuro, si può generare il dizionario personale relativo.

```
$ ispell -d ./minimo.hash -l < documento.txt > elenco [Invio]
```

In questo modo si ottiene l'elenco delle parole usate nel file '`documento.txt`', le quali sono praticamente tutte sconosciute. Questo elenco deve essere riordinato e ridotto.

```
$ sort -f < elenco > elenco1 [Invio]
```

```
$ munchlist -l minimo.aff -s minimo.hash elenco1 ↩
↪> dizionario [Invio]
```


Dopo la riduzione si ottiene finalmente il dizionario personale specifico del documento; successivamente si possono eseguire le verifiche sullo stesso documento di origine (a seguito di aggiunte o di modifiche), con il comando seguente:

```
$ ispell -d ./minimo.hash -p ./dizionario documento.txt [Invio]
```

47.9.2 Aspell

Aspell[24] è un programma funzionalmente simile a Ispell, che gestisce dei dizionari propri ed è in grado di utilizzare quelli appropriati alla configurazione locale:

```
aspell [opzioni] comando
```

I dizionari di Aspell dovrebbero trovarsi nella directory '`/usr/lib/aspell/`', assieme a dei file che gli consentono di riconoscere le codifiche più comuni dei caratteri. Vengono mostrati solo alcuni esempi di funzionamento del programma, senza descrivere le opzioni; per questo si può consultare la pagina di manuale *aspell(1)*, oppure la documentazione Info: *info aspell*.

- `$ aspell -c lettera` [*Invio*]

  Esamina il file '`lettera`', utilizzando il dizionario predefinito per la configurazione locale esistente. Se il programma incontra parole sconosciute, consente di intervenire in modo interattivo, come nella schermata seguente:

```
Ciao Antonio,
come stai?
È moltoa che non ci si sente; cosa fai di bello?
Daniele
-------------------------------------------------------------
1) molta                        2) molto


i) Ignora                       I) Ignora tutti
r) Rimpiazza                    R) Rimpiazza tutti
a) Aggiungi                     l) Aggiungi minuscolo
b) Abbandona                    x) Esci
-------------------------------------------------------------
?
```

  Come si vede, in questo caso la configurazione è quella italiana, dato che anche il menù appare tradotto. Il testo da verificare appare con qualche riga in più rispetto a Ispell, facilitando così il lavoro di correzione.

- `$ aspell --lang=it -c lettera` [*Invio*]

  Esamina il file '`lettera`', utilizzando espressamente il dizionario italiano.

- `$ aspell --encoding=utf-8 --lang=it -c lettera` [*Invio*]

  Esamina il file '`lettera`', utilizzando espressamente il dizionario italiano e la codifica UTF-8.

47.9.3 Myspell e Hunspell

Myspell è un componente di OpenOffice.org 1.∗ e di altri programmi, per la gestione dei dizionari ortografici. Hunspell[25] è un componente simile, nato per OpenOffice.org 2.∗, con la capacità di gestire meglio il riconoscimento delle parole composte.

In generale, comunque, si tratta sempre di lavori derivati o comunque ispirati al funzionamento di Ispell. In particolare, i dizionari realizzati per Myspell, sono utilizzabili anche da Hunspell.

Generalmente è possibile usare Hunspell anche attraverso un programma frontale simile a Ispell. Supponendo che i dizionari Myspell siano installati a partire da '`/usr/share/myspell/dicts/`', si potrebbe usare Hunspell come si vede nell'esempio seguente:

```
$ hunspell -d /usr/share/myspell/dicts/it-IT lettera [Invio]
```

Così facendo, il programma utilizza il dizionario composto dai file '`/usr/share/myspell/dicts/it-IT.aff`' e '`/usr/share/myspell/dicts/it-IT.dic`', per esaminare il file '`lettera`'.

47.10 Analisi sintattica e stilistica con Textchk

L'analisi sintattica di un testo è un problema ben più complicato della semplice verifica delle parole con un dizionario. Esistono però alcuni tipi di errori sintattici, o stilistici, che si possono identificare con l'aiuto di espressioni regolari (*regular expression*).

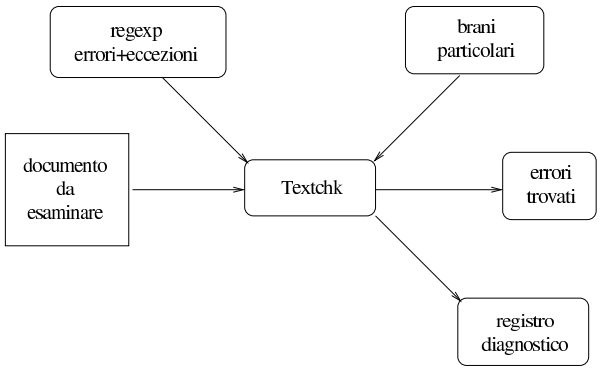
La lingua italiana consente spesso l'utilizzo di forme espressive differenti, per le quali dovrebbe esserci almeno uniformità all'interno di uno stesso documento. Per esempio, occorre decidere se si vuole scrivere: «una aula» oppure «un'aula», «ed anche» oppure «e anche»,...

In questa sezione si mostra un programma Perl che può aiutare a definire delle regole rappresentate in forma di espressioni regolari, per segnalare degli errori sintattici o stilistici. Con questo programma è possibile indicare anche delle regole di eccezione e delle particolarità riferite a un solo documento. Il programma in questione è Textchk: *allegati/textchk/textchk.pl* [26]

47.10.1 Principio di funzionamento

Textchk scandisce un file di partenza generando un altro file contenente le parti di testo che risulterebbero errate (oltre a un file diagnostico contenente la registrazione del procedimento di verifica). Prima di iniziare a leggere il file da esaminare, vengono caricati dei modelli che esprimono degli errori, espressi in forma di espressione regolare, seguiti eventualmente da dei modelli di eccezione. Infine, vengono caricate anche delle particolarità riferite al testo che si elabora, trattate in forma letterale e non più secondo il modello di un'espressione regolare.

Figura 47.69. Schema di funzionamento di Textchk.



Gli errori che si possono ricercare attraverso delle espressioni regolari, riguardano la vicinanza di parole che hanno caratteristiche determinate, come l'uso o meno di articoli apostrofati. Sotto questo aspetto, diventa importante che, nel file di testo originale, ogni paragrafo si trovi su una sola riga, cioè non sia interrotto su più righe.

A fianco di questo problema, si aggiunge il fatto che il file sorgente che si vuole esaminare potrebbe contenere dei codici di controllo, come nel caso di TeX (o LaTeX) e di HTML. In tutte queste situazioni, prima di passare all'analisi vera e propria, occorre ripulire e riadattare il testo, in modo da avere a che fare con un file di testo puro, in cui ogni paragrafo si trovi su una sola riga.

47.10.1.1 Espressioni regolari

Textchk è scritto in Perl, pertanto le espressioni regolari che possono essere gestite sono quelle di questo linguaggio di programmazione.

La ricerca della corrispondenza con le espressioni regolari che esprimono un errore, viene fatta in modo da circoscrivere, se possibile, tre parole prima e dopo della zona dell'errore. Per questa ragione, non ha senso tentare di identificare l'inizio e la fine di una riga (con i simboli '**^**' e '**$**'), inoltre non è possibile utilizzare le parentesi tonde.

A titolo di esempio, si propone il problema della «d» eufonica, per la precisione il caso di «ad». Supponendo di volerla utilizzare solo quando la parola successiva inizia con la vocale «a», escludendo il caso in cui la parola continui con un'altra «d» (per esempio: «ad amare», ma non «ad adattare»), si possono usare le espressioni regolari seguenti per individuare gli errori.

```
\ba[[:space:]]+a[^d][[:word:]]*\b
\bad[[:space:]]+ad[[:word:]]*\b
\bad[[:space:]]+[^a][[:word:]]*\b
```

Per intendere meglio il significato di ciò che è scritto, la prima riga significa:

'**\b**'	lo spazio vuoto prima della parola;
'**a**'	la lettera «a»;
'**[[:space:]]+**'	uno o più spazi orizzontali;
'**a[^d]**'	la lettera «a» seguita immediatamente da qualunque cosa che sia diversa dalla lettera «d»;
'**[[:word:]]***'	zero o più caratteri alfabetici;
'**\b**'	lo spazio vuoto dopo la parola.

Nello stesso tempo, però, si può decidere di accettare un'eccezione: «ad esempio», che secondo quanto stabilito con l'ultima delle espressioni regolari appena mostrate, dovrebbe essere un errore. Si può usare quindi l'espressione regolare seguente, tra le eccezioni.

```
\bad[[:space:]]+esempio\b
```

47.10.2 Configurazione

La configurazione di Textchk serve a definire gli errori sintattici che si ricercano. In generale è importante definire una configurazione specifica per ogni singolo progetto di documentazione, ma resta la possibilità di stabilire regole personali, legate all'utente, oltre che regole generali legate al sistema (per quanto questo possa avere un valore relativo).

La configurazione avviene attraverso un file di testo normale, in cui le righe bianche, quelle vuote e quelle che iniziano con il simbolo '**#**' vengono ignorate. Le altre righe sono dei record che possono avere una delle due forme seguenti:

```
DBL____regola_di_errore [____testo_esplicativo]
```

```
ERR____regola_di_errore [____testo_esplicativo]
```

```
EXC____regola_di_eccezione
```

Nel primo caso si identifica una parola che si ritiene possa essere stata scritta due volte, in modo erroneo; il secondo indica un modello di errore, mentre nel terzo si tratta di un'eccezione. I record che descrivono le regole di eccezione si riferiscono sempre all'ultima regola di errore (di tipo '**DBL**' o '**ERR**') che sia stata incontrata fino a quel punto.

La forma di questi record è un po' strana, nel senso che la separazione dei campi avviene attraverso una sequenza di quattro trattini bassi ('____'). Ciò serve per evitare di creare problemi alla realizzazione delle espressioni regolari che descrivono gli errori e le eccezioni.

```
#
# d eufonica
# a|e|o prendono una «d» eufonica se sono seguite da una
# parola che inizia con la stessa vocale, a meno che ci sia
# subito dopo un'altra «d».
#
ERR____\ba[[:space:]]+a[^d][[:word:]]*\b____a --> ad
EXC____\bda[[:space:]]+a[[:space:]]+a\b
#
ERR____\bad[[:space:]]+ad[[:word:]]*\b____ad --> a
#
ERR____\bad[[:space:]]+[^aA][[:word:]]*\b____ad --> a
EXC____\bad[[:space:]]+esempio\b
EXC____\bad[[:space:]]+ora\b
#
ERR____\be[[:space:]]+e[^d][[:word:]]*\b____e --> ed
ERR____\bed[[:space:]]+[eE]d[[:word:]]*\b____ed --> e
ERR____\bed[[:space:]]+[^eèE][[:word:]]*\b____ed --> e
#
```

```
ERR____\bo[[:space:]]+[oO][^d][[:word:]]*\b____o --> od
ERR____\bod[[:space:]]+[oO]d[[:word:]]*\b____od --> o
ERR____\bod[[:space:]]+[^oO][[:word:]]*\b____od --> o
```

L'esempio mostra una serie di istruzioni con le quali si cerca di definire l'uso della «d» eufonica. Vale la pena di analizzare cosa succede di fronte a una situazione precisa. Si suppone di avere scritto un testo nel quale è stata inserita la frase seguente, la quale risulta essere disposta su una sola riga:

```
Purtroppo, fino ad ora il colore dell'auto non è stato ↩
↪scelto dal cliente.
```

Concentrando l'attenzione sui record di configurazione seguenti, si può simulare ciò che succede.

```
ERR____\bad[[:space:]]+[^aA][[:word:]]*\b____ad --> a
EXC____\bad[[:space:]]+esempio\b
EXC____\bad[[:space:]]+ora\b
```

Per cominciare, viene individuato un errore in via preliminare in corrispondenza di «ad ora», perché la parola che segue «ad» non inizia con una lettera «a». Textchk preleva una stringa di tre parole prima e tre parole dopo questo errore: «Purtroppo, fino ad ora il colore dell'auto». In questo caso, le parole precedenti sono solo due, perché non è stato possibile ottenere di più.

Su questa stringa estratta viene condotto il controllo per le eccezioni successive; così, dal momento che si ottiene una corrispondenza (sempre con «ad ora»), l'errore si rivela infondato (in base ai presupposti stabiliti).

L'ultimo campo dei record che descrivono gli errori serve per indicare una spiegazione per ciò che viene identificato come un errore. Questa spiegazione viene mostrata da Textchk nel momento in cui l'errore relativo viene mostrato, secondo lo schema seguente:

testo_esplicativo
tre_parole_precedenti>>errore<<tre_parole_successive

Come si vede, la corrispondenza con l'errore viene evidenziata dai delimitatori '**>>**' e '**<<**'.

47.10.2.1 Gerarchia della configurazione

Textchk è stato pensato originariamente per avere una configurazione specifica per ogni progetto di documentazione che ogni autore possa gestire. Tuttavia, è possibile definire anche una configurazione personale e una di sistema. Si tratta dei file seguenti:

'`./.textchk.rules`'	contiene la configurazione corrente, che viene letta prima delle altre;
'`~/.textchk.rules`'	contiene la configurazione personale, letta subito dopo quella corrente;
'`/etc/textchk.rules`'	contiene la configurazione di sistema, che viene letta alla fine.

In generale non è opportuno stabilire una configurazione generale di sistema. Tuttavia, se c'è la necessità di annullare l'effetto di una regola di errore stabilita a livello generale, si può dichiarare la stessa regola nella configurazione personale o in quella corrente, facendola seguire immediatamente da un'eccezione identica. Per esempio, si può supporre di avere definito a livello di sistema la regola seguente, che richiede l'uso della «d» eufonica ogni volta che la parola seguente inizia con una vocale:

```
ERR____\b[aeo][[:space:]]+[aeiouAEIOU][[:word:]]*\b____a/e/o --> ad/ed/od
```

Per annullarne l'effetto completamente, basta aggiungere la stessa regola in qualità di eccezione, subito dopo:

```
#
# Regola di sistema che qui viene annullata.
#
ERR____\b[aeo][[:space:]]+[aeiouAEIOU][[:word:]]*\b____a/e/o --> ad/ed/od
EXC____\b[aeo][[:space:]]+[aeiouAEIOU][[:word:]]*\b
```


47.10.2.2 Casi particolari

Alle volte non conviene indicare troppe eccezioni, oppure non è materialmente possibile. Per esempio, si può immaginare il caso in cui si vuole mostrare veramente un modo sbagliato di scrivere per qualche ragione. Per queste situazioni viene in aiuto un file di configurazione aggiuntivo, che però può essere associato esclusivamente a un solo progetto di documentazione. Si tratta del file '`./.textchk.special`', in cui si possono inserire integralmente alcune stringhe che Textchk ha indicato precedentemente come errate.

Per questa parte della configurazione non c'è molto da fare: basta utilizzare un programma per la creazione e la modifica dei file di testo ricopiando ciò che serve dal file che viene generato da Textchk per registrare gli errori trovati. L'esempio seguente mostra un estratto di quello che potrebbe contenere questo file. Si osservi il fatto che si tratta di esempi di errori scritti così di proposito.

```
oppure «un'aula», «ed anche» oppure «e
vuole scrivere: «una aula» oppure «un'aula»,
ma non «ad adattare»), si possono
```

47.10.2.3 L'indicazione di parole doppie

Un errore frequente nella scrittura di un testo consiste nella ripetizione di una parola per due volte di seguito, mentre l'intenzione sarebbe quella di scriverla una volta sola. Per intercettare questo tipo di situazione si utilizza il record '**DBL**'. Nel campo dell'espressione che indica l'errore, si fa riferimento implicitamente a una parola intera. Infatti, nella comparazione reale, viene aggiunto il simbolo '**\b**' all'inizio e alla fine, a sottolineare che la parola deve essere completa. Si osservi l'esempio seguente:

```
#
# Parole doppie.
#
DBL____[[:word:]][[:word:]]+____Due parole identiche
EXC____\bciao[[:space:]]+ciao\b
```

L'intenzione è di individuare qualunque parola ('**[[:word:]]+**'), composta almeno da due caratteri, che si ripete immediatamente. Viene posta una sola eccezione alla coppia «ciao ciao».

47.10.3 Come si usa

Textchk si compone di un eseguibile unico, '**textchk**', che si utilizza secondo lo schema sintattico seguente:

```
textchk --input-type=tipo_di_file file_da_analizzare ↩
↪      [errori_risultanti [file_diagnostico]]
```

```
textchk --help
```

```
textchk --version
```

Oltre alle opzioni standard, '**--help**' e '**--version**', l'opzione '**--input-type**' serve a stabilire il tipo di file che si fornisce in ingresso, in modo che Textchk sappia come fare per gestirlo opportunamente, attraverso un argomento:

Opzione	Descrizione
`--input-type=standard`	si riferisce a un file di testo in cui ogni capoverso occupa esattamente una riga e non richiede altri adattamenti;
`--input-type=man`	si riferisce a un file Troff delle pagine di manuale;
`--input-type=texinfo` `--input-type=texi`	si riferisce a un sorgente Texinfo;
`--input-type=html`	si riferisce a un file HTML.

Il secondo argomento della riga di comando è il nome del file da

analizzare, che deve corrispondere al tipo indicato precedentemente. Il terzo argomento serve a definire il nome del file che viene creato per annotare le stringhe errate che vengono individuate; se non viene fornito espressamente il suo nome, viene creato un file con lo stesso nome di quello in ingresso, con l'aggiunta dell'estensione '`.tchk`' ('***file_da_analizzare*** `.tchk`'). Il quarto argomento serve a specificare il nome del file diagnostico, nel quale vengono registrate tutte le fasi di individuazione di errori e di eccezioni. Anche l'indicazione di questo file può essere omessa; in tal caso viene usato il nome del file degli errori con l'aggiunta dell'estensione '`.tdiag`', oppure il file in ingresso con la stessa aggiunta ('***errori_risultanti*** `.tdiag`' oppure '***file_da_analizzare*** `.tdiag`').

Per esempio, il comando seguente genera i file '`bash.1.tchk`' e '`bash.1.tdiag`':

```
$ textchk --input-type=man bash.1 [Invio]
```

47.10.3.1 Come vengono mostrati gli errori e i dati diagnostici

Durante il suo lavoro, Textchk mostra sullo schermo ciò che trova, delimitando gli errori tra i delimitatori '**>>**' e '**<<**'. Per esempio, in base alle regole seguenti,

```
ERR____\bad[[:space:]]+[^aA][[:word:]]*\b____ad --> a
EXC____\bad[[:space:]]+esempio\b
EXC____\bad[[:space:]]+ora\b
```

si possono ottenere segnalazioni come queste:

```
ad --> a
    Pertanto, andando >>ad elevare<< il proprio livello
ad --> a
    contrario, riuscendo così >>ad esplorare<< il proprio
```

Nel file che elenca gli errori si trovano le righe seguenti:

```
Pertanto, andando ad elevare il proprio livello
contrario, riuscendo così ad esplorare il proprio mondo
```

Inoltre, nel file diagnostico si trova l'intero procedimento:

```
??? Pertanto, andando >>ad elevare<< il proprio livello
ERR \bad[[:space:]]+[^aA][[:word:]]*\b
!!! Pertanto, andando >>ad elevare<< il proprio livello

??? contrario, riuscendo così >>ad esplorare<< il proprio mondo
ERR \bad[[:space:]]+[^aA][[:word:]]*\b
!!! contrario, riuscendo così >>ad esplorare<< il proprio mondo

??? Il colore rosso, >>ad esempio<<, rappresenta la propria
ERR \bad[[:space:]]+[^aA][[:word:]]*\b
EXC \bad[[:space:]]+esempio\b

??? Il colore rosso, >>ad esempio<<, rappresenta la propria
ERR \bad[[:space:]]+[^aA][[:word:]]*\b
EXC \bad[[:space:]]+esempio\b

??? Pertanto, l'espressione «>>ad emettere<<» non è corretta.
ERR \bad[[:space:]]+[^aA]][[:word:]]*\b
SPC Pertanto, l'espressione «ad emettere» non è corretta.
```

Il file diagnostico mostra informazioni diverse, distinte attraverso una sigla iniziale. Le righe che iniziano con '**???**' indicano il problema trovato; le righe che iniziano con '**ERR**' rappresentano la regola di errore in base alla quale viene evidenziato il problema; le righe che iniziano con '**EXC**' indicano una regola di eccezione per la quale il problema viene superato; le righe che iniziano con '**SPC**' rappresentano un caso particolare (speciale), per cui la frase in questione viene accettata così come si trova. Infine, le righe che iniziano con '**!!!**' rappresentano la conferma finale che si deve trattare di un errore.

47.10.4 Come si installa

Textchk si compone di un solo programma Perl: '**textchk**'. Questo file può essere collocato ovunque sia ritenuto più conveniente, preferendo evidentemente una directory elencata all'interno della variabile di ambiente ***PATH***.

Trattandosi di un programma Perl, deve essere disponibile l'interprete relativo. Attualmente si prevede che questo corrisponda esattamente all'eseguibile '`/usr/bin/perl`'. Se il proprio sistema non è organizzato in questo modo, basta modificare la prima parte del programma:

```
#!/usr/bin/perl
…
```

Dopo la soluzione di questo problema, c'è solo bisogno di predisporre un file di regole, '`./.textchk.rules`', poi, mano a mano che il lavoro procede, può essere conveniente predisporre anche il file '`./.textchk.special`'.

47.10.4.1 Gettext

I messaggi che può mostrare Textchk possono essere tradotti, dal momento che viene usato il modulo Perl-gettext. Nel pacchetto del sorgente è presente un file di messaggi per la lingua italiana, che però deve essere compilato e installato:

```
$ msgfmt -o textchk.mo it.po [Invio]
```

In questo modo, si genera il file '`textchk.mo`', che probabilmente va collocato nella directory '`/usr/share/locale/it/LC_MESSAGES/`'.

47.10.4.2 Dipendenze

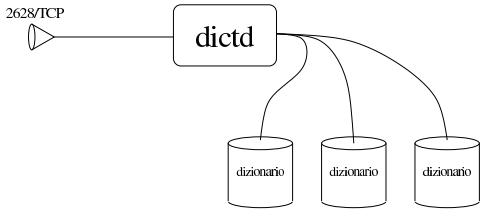
Per funzionare, Textchk richiede l'interprete Perl e la presenza di un modulo speciale: Perl-gettext. Inoltre, per poter gestire correttamente i diversi tipi di file per cui è stato predisposto, richiede in particolare Groff, Lynx e Texinfo.

47.11 Dizionari

Esistono diversi tipi di dizionario che spesso sono disponibili in forma di file di testo, redatti secondo un certo standard. Non si tratta esclusivamente di dizionari per le traduzioni, ma si possono includere quelli per i sinonimi, quelli specifici per un certo settore tecnico, fino a quelli biblici.

Per accedere a tutti i dizionari disponibili presso un elaboratore, si usa generalmente un programma demone, a cui si accede attraverso la rete, precisamente secondo il protocollo DICT (*Dictionary server protocol*) descritto dal documento RFC 2229. Naturalmente, i file contenenti i dizionari, devono essere realizzati secondo il formato atteso dal demone stesso. Di solito si considera il programma '**dictd**' per questo scopo.

Figura 47.87. Il servizio DICT viene fornito attraverso un demone, di solito in ascolto della porta 2628/TCP.



47.11.1 Dictd o Serpento

Il programma che di solito si usa come servente DICT è Dictd[27] oppure Serpento[28] (il secondo sembra avere una gestione migliore dell'insieme di caratteri universale).

```
dictd
```

```
serpento
```

Il programma, sia '**dictd**', sia '**serpento**', salvo indicazioni differenti nella sua configurazione, si collega ai file che dovrebbero trovarsi nella directory '`/usr/share/dictd/`' e si mette in ascolto della porta 2628/TCP.

I file in questione devono essere conformi a quanto atteso dai programmi. Nel caso della distribuzione GNU/Linux Debian, i dizionari sono distribuiti in pacchetti separati, il cui nome inizia per '**`dict-`**'.

47.11.2 Interrogazione manuale di un servizio DICT

Per comprendere l'utilità del servizio DICT, prima di avvicinarsi all'uso dei programmi clienti che possono servirsene, conviene provare a utilizzare direttamente il protocollo, con i suoi comandi principali. Ciò consente anche di verificare il funzionamento del servizio. Per l'interrogazione diretta, si può usare Netcat o Telnet:

```
netcat nodo 2628
```

```
telnet nodo 2628
```

Ecco come si potrebbe usare Telnet per accedere all'elaboratore locale, presso il quale è in funzione Serpento:

```
$ telnet localhost 2628 [Invio]

  Trying 127.0.0.1...
  Connected to 127.0.0.1.
  Escape character is '^]'.
  220 hello <> msg

SHOW SERVER [Invio]

  114 server information follows
  serpento
  .
  250 Command complete

QUIT [Invio]

  221 bye bye
  Connection closed by foreign host.
```

Prima di vedere i comandi più importanti, è necessario chiarire il significato di alcune definizioni che si rifanno a quanto scritto nel documento RFC 2229.

- Ogni dizionario viene considerato costituire una «base di dati», ma qui non si usa questa dizione che sembra impropria.
- La ricerca di un termine avviene attraverso un confronto, il cui metodo può cambiare in base alla necessità. Per esempio si può cercare una corrispondenza esatta, oppure una corrispondenza fonetica. Il metodo di confronto di una ricerca viene definito «strategia»; qui si fa riferimento al metodo di ricerca.

La tabella successiva mostra la sintassi, semplificata di alcuni comandi.

Sintassi	Descrizione
`DEFINE` ***dizionario parola*** `DEFINE` {`!`|`*`} ***parola***	Richiede di trovare la definizione o la traduzione della parola indicata, cercandola nel dizionario stabilito (si cerca una corrispondenza esatta e completa della parola indicata). Se al posto del dizionari si indica un punto esclamativo, si vuole ottenere il risultato nel primo dizionario annoverato dal servizio; se invece si utilizza un asterisco, si vogliono trovare le corrispondenze con tutti i dizionari disponibili.

Sintassi	Descrizione
`MATCH` ***dizionario metodo_ricerca parola*** `MATCH` {`!`|`*`} ***metodo_ricerca parola***	Richiede di trovare una corrispondenza della parola indicata, attraverso l'indicazione di un metodo di ricerca (o strategia). Se al posto del dizionari si indica un punto esclamativo, si vuole ottenere il risultato nel primo dizionario annoverato dal servizio; se invece si utilizza un asterisco, si vogliono trovare le corrispondenze con tutti i dizionari disponibili.
`MATCH` ***dizionario*** `exact` ***parola***	Richiede di trovare una corrispondenza esatta della parola, praticamente come avviene con il comando '**`DEFINE`**'.
`MATCH` ***dizionario*** `prefix` ***parola***	Richiede di trovare una corrispondenza iniziale della parola.
`MATCH` ***dizionario . parola***	Richiede di trovare una corrispondenza secondo il metodo predefinito del servente, che dovrebbe corrispondere al migliore disponibile.
`SHOW DB` `SHOW DATABASE`	Richiede di mostrare l'elenco dei dizionari disponibili.
`SHOW STRAT` `SHOW STRATEGIES`	Richiede di mostrare l'elenco dei metodi di ricerca disponibili (quelli utilizzabili con il comando '**`MATCH`**'.
`SHOW SERVER` `SHOW SERVER`	Richiede di mostrare il nome del servente.
`QUIT`	Conclude il collegamento.

Segue la descrizione di alcuni esempi, partendo dalle interrogazioni necessarie a conoscere quali sono i dizionari disponibili.

- **`SHOW DB`** [*Invio*]

```
  110 57 databases here
  …
  freedict-iri-eng "freedict-iri-eng"
  freedict-eng-lat "freedict-eng-lat"
  freedict-deu-ita "freedict-deu-ita"
  freedict-deu-por "freedict-deu-por"
  .
  250 Command complete
```

- **`SHOW STRAT`** [*Invio*]

```
  111 9 strategies here
  soundex "Match using SOUNDEX algorithm"
  suffix "Match suffixes"
  metaphone "metaphone algorithm"
  substring "Match substring occurring anywhere in word"
  re "POSIX 1003.2 regular expressions"
  prefix "Match prefixes"
  lev "Match words within Levenshtein distance one"
  fnmatch "fnmatch-like (* ? as wildcards)"
  exact "Match words exactly"
  .
  250 Command complete
```

  I metodi di ricerca '**`exact`**' e '**`prefix`**' sono obbligatori, nel senso che devono essere sempre presenti, secondo quanto stabilito dal documento RFC 2229.

- **`DEFINE ! ciao`** [*Invio*]

  In questo caso si chiede la definizione (la corrispondenza esatta) della parola «ciao» con il primo dizionario disponibile:

```
150 1 here you are
151 "ciao" freedict-eng-cro "freedict-eng-cro"
ciao

ćao
.
250 Command complete
```

In questo caso è stata ottenuta la traduzione dall'inglese al croato.

- **`DEFINE * ciao`** [*Invio*]

  Questa volta si vuole la definizione della parola da tutti i dizionari disponibili; qui si vede solo l'ultimo:

```
150 5 here you are
…
151 "ciao" freedict-ita-eng "freedict-ita-eng"
ciao
     hello
.
250 Command complete
```

- **`MATCH ! soundex amore`** [*Invio*]

  Viene cercata la parola «amore», attraverso il metodo di ricerca (strategia) '**`soundex`**', limitando l'interesse al primo dizionario che contenga almeno una corrispondenza valida:

```
152 9 here you are
…
freedict-fra-eng "amour"
freedict-fra-eng "aumônier"
freedict-fra-eng "annuaire"
freedict-fra-eng "amener"
.
250 Command complete
```

Eventualmente, dopo si può fare una ricerca con il comando '**`DEFINE`**', specificando anche il dizionario, per avere maggiori indicazioni su una parola dell'elenco trovato.

47.11.3 Il programma Dict per l'interrogazione del servizio

Dict[29] è il programma più comune per l'interrogazione di un servizio DICT. Si usa a riga di comando, ma proprio per questo è molto semplice ed efficace:

```
dict [opzioni] parola
```

```
dict [opzioni] dict://nodo/d:parola[:dizionario]
```

```
dict [opzioni] dict://nodo/m:parola↩
↪[:dizionario[:modalità_di_ricerca]]
```

La prima forma di utilizzo del programma implica la ricerca della parola indicata presso il servente o i serventi indicati nella configurazione (salvo indicazione diversa attraverso le opzioni); la seconda richiede una ricerca presso un elaboratore stabilito, utilizzando un confronto completo (l'equivalente del comando '**`DEFINE`**' del protocollo DICT); l'ultima richiede una ricerca presso un elaboratore stabilito, utilizzando un confronto che non sia necessariamente esatto (l'equivalente del comando '**`MATCH`**' del protocollo DICT).

Il file di configurazione generale di Dict può essere, a seconda dei sistemi, '`/etc/dict.conf`', oppure '`/etc/dictd/dict.conf`'; il file di configurazione personale di ogni utente è '`~/.dictrc`'. La configurazione serve a specificare i serventi predefiniti, aggiungendo eventualmente indicazioni sulla porta TCP da utilizzare, nel caso non sia quella standard:

```
server nodo [port n_porta]
```

Eventualmente sono disponibili altre opzioni nella direttiva '**`server`**' nei casi in cui l'accesso al servente richieda una forma di autenticazione. È normale trovare queste due direttive nel file di configurazione:

```
server localhost
server dict.org
```

In pratica, si cerca prima un servizio DICT localmente, altrimenti si interroga quello fornito da *dict.org*.

Tabella 47.98. Alcune opzioni per l'utilizzo di Dict.

Opzione	Descrizione
`-h` ***nodo*** `--host` ***nodo***	Specifica di interrogare un servente particolare, che probabilmente è diverso da quanto indicato nella configurazione.
`-p` ***n_porta*** `--port` ***n_porta***	Specifica di rivolgersi a un certo numero di porta TCP, evidentemente diverso da quello predefinito che sarebbe 2628.
`-d` ***dizionario*** `--database` ***dizionario***	Specifica il dizionario all'interno del quale si vuole eseguire la ricerca.
`-D` `--dbs`	Richiede un elenco dei dizionari disponibili presso il servente.
`-m` `--match`	Invece di ottenere una «definizione», si richiede di cercare una corrispondenza (sulla base del metodo di ricerca specificato con l'opzione '**`--strategy`**').
`-s` ***metodo_ricerca*** `--strategy` ***metodo_ricerca***	Se si usa l'opzione '**`-m`**', consente di specificare il metodo di ricerca.
`-S` `--strats`	Richiede un elenco dei metodi di ricerca (strategie) disponibili presso il servente.

Segue la descrizione di alcuni esempi.

- **`$ dict -D`** [*Invio*]

```
Databases available:
…
 freedict-iri-eng freedict-iri-eng
 freedict-eng-lat freedict-eng-lat
 gazetteer2k-places gazetteer2k-places
 freedict-deu-ita freedict-deu-ita
 freedict-por-deu freedict-por-deu
 freedict-deu-por freedict-deu-por
```

- **`$ dict -S`** [*Invio*]

```
Strategies available:
 soundex    Match using SOUNDEX algorithm
 suffix     Match suffixes
 metaphone  metaphone algorithm
 substring  Match substring occurring anywhere in word
 re         POSIX 1003.2 regular expressions
 prefix     Match prefixes
 lev        Match words within Levenshtein distance one
 fnmatch    fnmatch-like (* ? as wildcards)
 exact      Match words exactly
```

I metodi di ricerca '**`exact`**' e '**`prefix`**' sono obbligatori, nel senso che devono essere sempre presenti, secondo quanto stabilito dal documento RFC 2229.

- **`$ dict \! ciao`** [*Invio*]

  In questo caso si chiede la definizione (la corrispondenza esatta) della parola «ciao» con il primo dizionario disponibile. Il punto esclamativo appare preceduto dalla barra obliqua inversa perché altrimenti la shell lo interpreterebbe con un significato speciale.

```
1 definition found

From freedict-eng-cro [freedict-eng-cro]:

  ciao

  ćao
```

In questo caso è stata ottenuta la traduzione dall'inglese al croato.

- `$ dict -d \* ciao` [*Invio*]

Questa volta si vuole la definizione della parola da tutti i dizionari disponibili (anche in questo caso la barra obliqua inversa, che precede l'asterisco, serve a evitare che la shell interpreti l'asterisco in modo speciale); qui si vede soltanto l'ultimo:

```
6 definitions found
…
From freedict-ita-eng [freedict-ita-eng]:

  ciao
       hello
```

- `$ dict ciao` [*Invio*]

Si ottiene esattamente la stessa cosa del comando precedente; poiché, se non si specifica il dizionario, vengono scanditi tutti.

- `$ dict -m -s soundex amore` [*Invio*]

Viene cercata la parola «amore», attraverso il metodo di ricerca (strategia) '**soundex**', su tutti i dizionari disponibili:

```
gcide:  Aimer  Ameer  Amir  Amorwe  Amour  Anear  Annoyer
  Annuary  Anoura  anoura  Anura  anuria  Anury  Anywhere
  Aumery
…
foldoc:  annoyware  anr
easton:  Amariah  Aner
hitchcock:  Amariah  Aner
gazetteer:  Amery  Amory  Anmoore  Aynor
gaz-place:  Amery  "Amery, WI"  Amory  Anmoore  Aynor
```

Eventualmente, dopo si può fare una ricerca normale (come «definizione»), specificando anche il dizionario, per avere maggiori indicazioni su una parola dell'elenco trovato.

- `$ dict -D -h dict.org` [*Invio*]

Elenca i dizionari disponibili presso l'elaboratore *dict.org*:

```
Databases available:
…
 english    English Monolingual Dictionaries
 trans      Translating Dictionaries
 all        All Dictionaries (English-Only and Translating)
 web1913    Webster's Revised Unabridged Dictionary (1913)
 world95    The CIA World Factbook (1995)
```

- `$ dict -S -h dict.org` [*Invio*]

```
Strategies available:
 exact      Match headwords exactly
 prefix     Match prefixes
 substring  Match substring occurring anywhere in a headword
 suffix     Match suffixes
 re         POSIX 1003.2 (modern) regular expressions
 regexp     Old (basic) regular expressions
 soundex    Match using SOUNDEX algorithm
 lev        Match headwords within Levenshtein distance one
 word       Match separate words within headwords
```

- `$ dict dict://dict.org/d:ciao` [*Invio*]

Si chiede la definizione (la corrispondenza esatta) della parola «ciao» con tutti i dizionari disponibili presso *dict.org*.

```
1 definition found

From WordNet (r) 2.0 [wn]:

  ciao
       n : an acknowledgment that can be used to say hello or
           goodbye (aloha is Hawaiian and ciao is Italian)
           [syn: {aloha}]
```

- `$ dict dict://dict.org/d:love:easton` [*Invio*]

Cerca la definizione della parola «love» nel dizionario '**easton**', presso *dict.org*:


```
1 definition found

From Easton's 1897 Bible Dictionary [easton]:

  Love
     This word seems to require explanation only in the case
     of its use by our Lord in his interview with "Simon,
     the son of Jonas,"
…
       In 1 Cor. 13 the apostle sets forth the excellency of
     love, as the word "charity" there is rendered in the
     Revised Version.
```

- `$ dict dict://dict.org/m:amore:\*:soundex` [*Invio*]

Viene cercata la parola «amore», attraverso il metodo di ricerca (strategia) '**soundex**', su tutti i dizionari disponibili, presso *dict.org*:

```
gcide:  Aimer  Ameer  Amir  Amorwe  Amour  Anear  Annoyer
  Annuary  Anoura  anoura  Anura  anuria  Anury  Anywhere
  Aumery
…
hitchcock:  Amariah  Aner
gazetteer:  Amery  Amory  Anmoore  Aynor
gaz-place:  Amery  "Amery, WI"  Amory  Anmoore  Aynor
```

47.12 Riferimenti

- Markus Kuhn, *International standard paper sizes*, http://www.cl.cam.ac.uk/~mgk25/iso-paper.html
- R. Smith, F. Wright, T. Hastings, S. Zilles, J. Gyllenskog, *RFC 1759: Printer MIB, Appendix B - Media size names from ISO/IEC 10175 Document printing architecture*, 1995, http://www.ietf.org/rfc/rfc1759.txt
- T. Hastings, R. Herriot, R. deBry, S. Isaacson, P. Powell, *RFC 2911: Internet printing protocol/1.1: model and semantics, Appendix C: "media" keyword values*, 2000, http://www.ietf.org/rfc/rfc2911.txt
- *Grafica; scienza, tecnologia e arte della stampa e della comunicazione*, Arti poligrafiche europee
- Claudio Beccari, *LaTeX, Guida a un sistema di editoria elettronica*, Hoepli, 1991, ISBN 88-203-1931-4
- M. Fazio, *Dizionario e manuale delle unità di misura*, Zanichelli
- Marco Gaiarin, *Linux Italian HOWTO*, http://oss.sgi.com/LDP/HOWTO/Italian-HOWTO.html
- Maurizio Pistone, *Lingua italiana e altra linguistica*, http://www.mauriziopistone.it/testi/linguaitaliana.html
- Marco Baroni, Eros Zanchetta, *Morph-it! A free morphological lexicon for the Italian Language*, http://sslmitdev-online.sslmit.unibo.it/linguistics/morph-it.php
- *Amiga Translators' Organization*, http://bilbo.di.unipi.it/~ato-it/
- Bureau International des Poids et Mesures, http://www.bipm.org/
- Bureau International des Poids et Mesures, *Le Système international d'unités (SI)*, http://www1.bipm.org/utils/en/pdf/brochure-si.pdf
- Bureau International des Poids et Mesures, *The International System of Units (SI)* (traduzione in inglese), http://www1.bipm.org/utils/en/pdf/si-brochure.pdf
- National Institute of Standards and Technology, *International System of Units (SI)*, http://physics.nist.gov/Pubs/SP330/sp330sl30.pdf
- National Institute of Standards and Technology, *Guide for the Use of the International System of Units (SI)*, 1995, http://physics.nist.gov/cuu/pdf/sp811.pdf

- Markus Kuhn, *Standardized Units for Use in Information Technology*, 1995, http://www.cl.cam.ac.uk/~mgk25/information-units.txt
- National Institute of Standards and Technology, *Prefixes for binary multiples*, http://physics.nist.gov/cuu/Units/binary.html
- *OneLook dictionary search*, http://www.onelook.com/
- Denis Howe, *FOLDOC: free on-line dictionary of computing*, http://foldoc.org/
- Jukka Korpela, *A tutorial on character code issue*, http://www.cs.tut.fi/~jkorpela/chars.html
- *Unicode Home Page*, http://www.unicode.org/
- *Frequently asked questions*, http://www.unicode.org/faq/
- Ken Whistler, Mark Davis, *Unicode Technical Report #17, Character Encoding Model*, http://www.unicode.org/reports/tr17/
- C. Weider, C. Preston, K. Simonsen, H. Alvestrand, R. Atkinson, M. Crispin, P. Svanberg, *RFC 2130: The Report of the IAB Character Set Workshop held 29 February - 1 March, 1996*, http://www.ietf.org/rfc/rfc2130.txt
- Markus Kuhn, *UTF-8 and Unicode FAQ for UNIX/Linux*, http://www.cl.cam.ac.uk/~mgk25/unicode.html
- Bruno Haible, *The Unicode HOWTO*, http://tldp.org/HOWTO/Unicode-HOWTO.html
- D. Goldsmith, M. Davis, *RFC 2152: UTF-7, a mail-safe transformation format of Unicode*, 1997, http://www.ietf.org/rfc/rfc2152.txt
- F. Yergeau, *RFC 2279: UTF-8, a transformation format of ISO 10646*, 1998, http://www.ietf.org/rfc/rfc2279.txt
- Indrek Hein, *An online character database*, http://www.eki.ee/letter/
- Jukka Korpela, *The ISO Latin 1 character repertoire - a description with usage and notes*, http://www.cs.tut.fi/~jkorpela/latin1/
- Roman Czyborra, *The ISO 8859 Alphabet Soup*, http://czyborra.com/charsets/iso8859.html
- Roman Czyborra, *Codepages & Co.*, http://czyborra.com/charsets/codepages.html
- *Character Sets*, http://utopia.knoware.nl/users/eprebel/Communication/CharacterSets/
- IBM, *National language support guide and reference*, http://publib.boulder.ibm.com/infocenter/pseries/v5r3/topic/com.ibm.aix.nls/doc/nlsgdrf/nlsgdrf.pdf
- *Unicode character database*, http://www.unicode.org/Public/UNIDATA/
- *DICT.org*, http://www.dict.org
- *RFC 2229: A dictionary server protocol*, 1997, http://www.ietf.org/rfc/rfc2229.txt

[1] Questi tre stili sono molto importanti, in parte per motivi storici, ma soprattutto perché sono quelli che si hanno a disposizione più di frequente.

[2] Alcuni sistemi di composizione riescono a trarre il corsivo e il neretto da stili che per loro natura non hanno tali varianti. Per ottenerlo si utilizzano tecniche di deformazione e di trascinamento. In generale sarebbe bene evitare di sfruttare tali possibilità, dal momento che se uno stile non dispone di una serie, significa che non è adatto per quella.

[3] La tipizzazione di un carattere è l'azione con cui lo si rende uniforme.

[4] Questa sezione va considerata solo come un riferimento essenziale alla definizione di uno stile letterario e il contenitore di una piccola raccolta di regole, con le quali semplificare la vita di chi scrive documenti elettronici. Pur senza una competenza specifica sulla questione, viene introdotto l'argomento sottolineando alcuni concetti importanti.

[5] Secondo una regola della tipografia del passato, ormai condannata generalmente, è necessario aumentare lo spazio che divide la fine di un periodo dall'inizio del successivo. Per qualche ragione si trovano ancora documenti in lingua inglese che seguono questa regola, anche quando si tratta di file di testo.

[6] Purtroppo LaTeX segue la vecchia regola dell'allungamento dello spazio dopo il punto fermo che chiude il periodo, con l'aggravante che per riuscire a determinarlo può fare solo delle supposizioni, che a volte sono errate. Per fare in modo che LaTeX eviti di applicare questa regola errata, si può utilizzare il comando '`\frenchspacing`' nel preambolo del documento.

[7] Quando il sistema di composizione si basa su TeX e si usano virgolette elevate, le virgolette doppie si ottengono preferibilmente attraverso una coppia di apici singoli aperti ('`‘‘`') e una coppia di apici singoli chiusi ('`’’`'). In altri casi, soprattutto quando si tratta di file di testo puri e semplici, gli apici doppi si indicano con le virgolette normali ('`"`…`"`').

[8] TeX permette l'uso di tre trattini di lunghezza differente: il trattino corto che si ottiene con un trattino singolo, il trattino medio che si ottiene con due trattini in sequenza e il trattino lungo che si ottiene con tre. Nella lingua italiana vanno usati solo i primi due, dove il trattino medio di TeX corrisponde al trattino lungo della grammatica italiana.

[9] Nell'ambito della documentazione tecnica, sarebbe consigliabile di evitare l'uso di accentazioni non comuni, anche se queste potrebbero essere preferibili in contesti più raffinati.

[10] Naturalmente questo ha senso se poi il programma di composizione non tenta di suddividere le parole in sillabe.

[11] «Ciò che si vede è ciò che si ottiene»

[12] Trattando di oggetti grafici astratti, sembra più opportuno parlare di «simboli», piuttosto che di segni; in quanto è la composizione a trasformarli in segni.

[13] Nel linguaggio C, si distingue tra un tipo «carattere» tradizionale, corrispondente al byte, e un tipo ***carattere esteso***, ovvero *wide char*. Di conseguenza ci sono sia stringhe tradizionali, sia ***stringhe estese***, ovvero *wide string*.

[14] In generale, per maggiore chiarezza, i punti di codifica dell'Unicode e di ISO 10646 si indicano nella forma '`U+`***`nnnn`***', oppure '`U-`***`nnnnnnnn`***', dove ***`n`*** è una cifra esadecimale; ma come è già stato mostrato, qui si usa la notazione '`#x`***`n`***' dell'XML.

[15] **Ascii** GNU GPL

[16] **Unicode** GNU GPL

[17] **Gucharmap** GNU GPL

[18] **Recode** GNU GPL

[19] **Iconv** GNU GPL

[20] **Ispell** software libero con licenza speciale

[21] Quando si ha a che fare con documentazione tecnica, in cui l'uso di termini in inglese è frequente, si potrebbe addirittura valutare la possibilità di basare l'analisi sul dizionario standard ('`english.hash`'), affiancando il dizionario personale specifico per il documento, solo che in tal caso si avrebbero difficoltà con le lettere accentate, dal momento che queste non sono previste nel file *affix* inglese.

[22] Le lettere '**ÿ**' e '**ß**', corrispondenti ai codici '`\377`' e '`\337`', sono minuscole e non hanno un equivalente maiuscolo nella codifica ISO 8859-1.

[23] In pratica, '**munchlist**' elimina queste parole ritenute estranee. Se si dispone di un elaboratore ben equipaggiato, si può dare in pasto a Ispell il file ottenuto dopo il riordino; lasciando a Ispell stesso il compito di eliminare i doppioni.

[24] **Aspell** GNU LGPL

[25] **Hunspell** GNU GPL, o GNU LGPL, o MPL

[26] **Textchk** GNU GPL

[27] **Dictd** GNU GPL

[28] **Serpento** GNU GPL

[29] **Dict** GNU GPL

PostScript: un linguaggio per la composizione finale

Benché il linguaggio PostScript sia nato per le stampanti, disponendo di Ghostscript potrebbe essere usato anche per scrivere direttamente. Lo scopo di questo capitolo è quello di introdurre all'uso diretto del linguaggio PostScript, in modo molto semplice, facendo riferimento prevalentemente al livello 1, eventualmente con qualche annotazione sul livello 2, mentre il livello 3 diventa troppo complesso e non più utile per un utilizzo diretto.

A ogni modo, gli esempi che si fanno sono stati verificati con Ghostscript e non con una vera stampante PostScript.

48.1 Impostazione generale

Un file PostScript è un file di testo, in cui le righe sono terminate indifferentemente con <*LF*> oppure con <*CR*><*LF*>, file che inizia con una riga simile al modello seguente:

```
%!PS-Adobe-livello_ps [EPSF-livello_eps]
```

Il livello è in pratica il numero di versione del linguaggio; per quanto riguarda il livello PostScript, si fa riferimento generalmente ai valori 1.0, 2.0 e 3.0. Il modello sintattico mostra la possibilità di aggiungere la stringa '**EPSF-*livello_eps***', con la quale si vuole specificare

che si tratta di un file PostScript incapsulato. In altri termini, un file PostScript normale inizia più o meno come nell'esempio seguente, dove si fa riferimento al livello 2:

```
%!PS-Adobe-2.0
…
```

In tal caso si intende lavorare su una serie di pagine; al contrario, se si sta realizzando una sola immagine nell'ambito di uno spazio determinato, si aggiunge la dichiarazione del tipo incapsulato:

```
%!PS-Adobe-2.0 EPSF-1.2
…
```

In generale, il simbolo di percentuale ('**%**') serve a introdurre dei commenti che non generano un risultato nella stampa; tuttavia, una sequenza di due simboli di percentuale ha un ruolo speciale per la dichiarazione di direttive importanti; inoltre, la stessa dichiarazione iniziale del tipo di file è preceduta da un simbolo percentuale. In generale, onde evitare equivoci, si indica un commento con un solo simbolo di percentuale seguito da almeno uno spazio:

```
% testo_commentato
```

Il commento può essere piazzato ovunque, tenendo presente che vale dal punto in cui appare, fino alla fine della riga.

Le direttive particolari che iniziano con due simboli di percentuale hanno la forma seguente:

```
%%direttiva [ : argomenti ]
```

In pratica, il nome delle direttive deve essere attaccato ai segni di percentuale; inoltre, se è prevista l'aggiunta di argomenti alla direttiva, dopo il nome della stessa appaiono due punti, seguiti da almeno uno spazio e dopo dagli argomenti previsti. Quello che segue è l'esempio di una struttura possibile per un file PostScript articolato su più pagine:

```
%!PS-Adobe-2.0
%%Creator: nome_del_redattore_del_file
%%DocumentPaperSizes: formato
%%EndComments
[%%BeginProlog
  prologo
%%EndProlog]
%%Page: numero_mostrato pagina_reale
istruzioni_ps
showpage
[%%Page: numero_mostrato pagina_reale
istruzioni_ps
showpage]…
%%Trailer
%%EOF
```

Il tutto dovrebbe essere abbastanza intuitivo: le prime istruzioni speciali, fino a '**%%EndComments**', descrivono il documento specificando in particolare le dimensioni della pagina; le istruzioni racchiuse tra '**%%BeginProlog**' e '**%%EndProlog**' possono servire per dichiarare delle funzioni utilizzate nel documento; le istruzioni relative a ogni singola pagina sono introdotte da '**%%Page: *m n***'; la visualizzazione della pagina, dopo la sua costruzione, si ottiene con l'istruzione '**showpage**'; il file termina con '**%%Trailer**' e '**%%EOF**'.

A questo punto, conviene vedere subito come si può articolare un file PostScript che non contiene pagine, ma una sola immagine:


```
%!PS-Adobe-2.0 EPSF-1.2
%%Creator: nome_del_redattore_del_file
%%BoundingBox: 0 0 larghezza_in_punti altezza_in_punti
%%EndComments
[%%BeginProlog
  prologo
%%EndProlog]
istruzioni_ps
showpage
%%Trailer
%%EOF
```

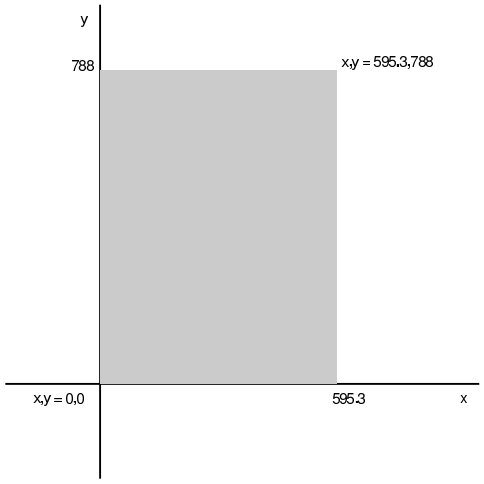
Si può osservare che la direttiva speciale '**%%BoundingBox**' va a sostituire '**%%DocumentPaperSizes**', allo scopo di indicare l'area in cui vanno rappresentate le istruzioni.

A ogni modo, la direttiva '**%%BoundingBox**' può essere usata anche per un file PostScript diviso in pagine, quando si vuole indicare un formato non standard, oppure se si vuole essere precisi.

48.1.1 Piano di lavoro e unità di misura

Il linguaggio PostScript è predisposto per fare riferimento a oggetti su un piano cartesiano ideale, in cui l'unità di misura normale è il punto tipografico, corrispondente secondo questo linguaggio a 1/72-esimo di pollice, pari a circa 0,35278 mm. In condizioni normali, la pagina è collocata sul piano cartesiano ideale come si vede nella figura 48.3, dove lo zero per ***x*** e ***y*** corrisponde esattamente con l'angolo inferiore sinistro; tuttavia, è consentita la definizione di pagine collocate in posizioni differenti, se questo può servire in qualche modo.

Figura 48.3. La pagina collocata su degli assi cartesiani ideali.



Gli oggetti grafici vengono disegnati sul piano cartesiano ideale; quello che risulta trovarsi sull'area della pagina può essere stampato.

48.1.2 Posizione corrente

Il PostScript può essere visto come un linguaggio per disegnare (tracciare curve, riempire delle aree e piazzare dei caratteri tipografici). Tutto questo avviene quasi sempre indicando delle coordinate, dove spesso la posizione di partenza ha importanza. Le coordinate iniziali si modificano con l'istruzione '**moveto**':

```
x y moveto
```

In pratica, si indicano due numeri, seguiti dalla parola chiave '**moveto**'. Il significato è molto semplice: il primo numero esprime la coordinata orizzontale (asse X), il secondo la coordinata verticale (asse Y). I valori si esprimono in punti tipografici.

48.1.3 Istruzioni speciali più importanti

Le istruzioni più importanti che iniziano con due segni di percentuale sono elencate brevemente nella tabella 48.4. Tuttavia, è il caso di aggiungere qualche piccola indicazione a proposito di alcune di queste.

Nessuna delle istruzioni che iniziano con due segni di percentuale è indispensabile; tuttavia alcune sono importanti. L'inserimento corretto di queste istruzioni rende il file PostScript più facile da gestire con gli strumenti comuni.

Tabella 48.4. Alcune istruzioni che iniziano con due segni di percentuale.

Istruzione	Descrizione
`%%Creator:` ***nome***	Il nome del programma che ha composto il file.
`%%DocumentPaperSizes:` ***formato***	Formato della carta.
`%%BoundingBox:` ***x_1 y_1 x_2 y_2***	Collocazione e dimensione della carta: da ***x_1***,***y_1*** a ***x_2***,***y_2***.
`%%Title:` ***titolo***	Titolo del documento.
`%%CreationDate:` ***data***	Data e ora di creazione del documento.
`%%Pages:` ***n***	Quantità di pagine contenuta.
`%%PageOrder: Ascend`|`Descend`	Ordine di apparizione delle pagine: ascendente o discendente.
`%%EndComments`	Fine dell'intestazione con le informazioni generali.
`%%BeginProlog`	Inizia un'area di definizione delle funzioni.
`%%EndProlog`	Termina l'area di definizione delle funzioni.
`%%BeginSetup`	Inizia un'area per l'inserimento di istruzioni di stampa.
`%%EndSetup`	Termina l'area delle istruzioni di stampa.
`%%Page:` ***x n***	Inizia la pagina ***n***-esima, rappresentata come ***x***.
`%%Trailer`	Conclude la serie delle pagine.
`%%EOF`	Conclude definitivamente il file.

L'istruzione '**`%%DocumentPaperSizes`**' serve intuitivamente per elencare le dimensioni possibili delle pagine. In generale si indica una sola parola chiave che esprime sinteticamente la dimensione della pagina, come si vede nella tabella 48.5. Probabilmente non conviene andare al di fuori di pochi standard; eventualmente è preferibile indicare le coordinate esatte attraverso l'istruzione '**`%%BoundingBox`**'.

Tabella 48.5. Formati di stampa comuni, indicabili come argomento dell'istruzione '**`%%DocumentPaperSizes`**'. Le dimensioni non sono necessariamente quelle reali, ma quelle conosciute dal linguaggio PostScript.

formato	larghezza punti	altezza punti	larghezza pollici	altezza pollici	larghezza centimetri	altezza centimetri
letter	612	792	8,50	11,00	21,59	27,94
legal	612	1008	8,50	14,00	21,59	35,56
a3	842	1190	11,6944	16,5278	29,7	42
a4	595	842	8,26389	11,6944	21	29,7
a5	421	595	5,84722	8,26389	14,85	21
b4	709	1002	9,84722	13,9167	25,0119	35,3483
b5	501	709	6,95833	9,84722	17,6742	25,0119

'**`%%BoundingBox`**' consente di indicare la posizione dell'angolo inferiore sinistro e di quello superiore destro della pagina. Di solito, le prime due coordinate che esprimono proprio la posizione dell'angolo inferiore sinistro, sono azzerate, a indicare che si parte dallo zero degli assi cartesiani ideali della superficie.

48.1.4 Aspetto delle istruzioni normali

Le istruzioni PostScript sembrano non avere inizio e fine, perché si possono collocare su una o più righe indifferentemente, senza alcun segno di separazione. Per esempio, si può scrivere:

```
newpath
 100   100 moveto
 100   431 lineto
 350   431 lineto
 350   100 lineto
closepath
```

Oppure, indifferentemente:

```
newpath 100 100 moveto 100 431 lineto 350 431 lineto ↩
↪350 100 lineto closepath
```

Naturalmente sono ammissibili tanti altri modi intermedi; comunque, è evidente che se si vuole scrivere del codice intelligibile occorre uno stile (come in tutti i linguaggi di programmazione).

La cosa che può apparire strana inizialmente è il fatto che i comandi che prevedono l'uso di argomenti, ricevono questi dati prima del nome del comando stesso. Per esempio, è già stata mostrata l'istruzione '**`moveto`**', la quale riceve l'indicazione delle coordinate prima del suo nome.

48.2 Linee e aree

La cosa più semplice che si può fare per cominciare a comprendere il linguaggio è quella di disegnare delle linee. In generale, il disegno avviene partendo dalle coordinate correnti, per cui questa indicazione non appare in modo esplicito, ma se necessario si definisce con uno spostamento attraverso l'istruzione '**`moveto`**'. Per le linee rette si possono usano le istruzioni '**`lineto`**' e '**`rlineto`**', dove la prima rappresenta un movimento con coordinate di destinazione assolute, mentre la seconda fa riferimento a coordinate di destinazione relative a quelle di partenza. Si osservino gli esempi seguenti, con cui si disegna lo stesso rettangolo largo 20 punti e alto 10 punti, a partire dalla coordinata ***x*** , ***y***=0 , 0:

```
  0    0 moveto
  0   10 lineto
 20   10 lineto
 20    0 lineto
  0    0 lineto
```

```
  0    0 moveto
  0   10 rlineto
 20    0 rlineto
  0  -10 rlineto
-20    0 rlineto
```

In pratica, l'istruzione '**`lineto`**' vuole l'indicazione del punto finale espresso come coordinata assoluta, mentre '**`rlineto`**' vuole una coordinata relativa alla posizione corrente.

Figura 48.10. Il rettangolo viene disegnato inserendo coordinate assolute, partendo dall'angolo inferiore sinistro e continuando con gli angoli successivi in senso orario.

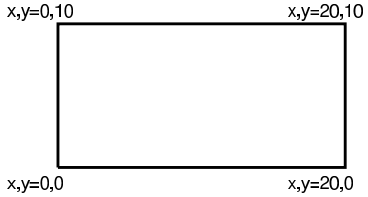
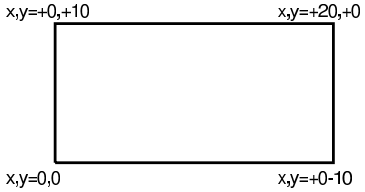


Figura 48.11. Il rettangolo viene disegnato inserendo coordinate relative.



Il disegno non viene tracciato se alla fine non si aggiunge un'istruzione '**stroke**', la quale non richiede argomenti. Dopo un'istruzione '**stroke**' viene perduto il riferimento alle coordinate correnti, per cui, se necessario, si deve ricominciare con un'istruzione '**moveto**'. Si osservi l'esempio seguente, in cui lo stesso rettangolo viene disegnato un segmento alla volta, riposizionando sempre le coordinate iniziali:

```
   0     0 moveto
   0    10 lineto stroke
   0    10 moveto
  20    10 lineto stroke
  20    10 moveto
  20     0 lineto stroke
  20     0 moveto
   0     0 lineto stroke
```

Naturalmente, se si preferisce questo modo di utilizzo dell'istruzione '**stroke**', si può anche cambiare un po' lo stile di scrittura:

```
   0     0 moveto    0    10 lineto stroke
   0    10 moveto   20    10 lineto stroke
  20    10 moveto   20     0 lineto stroke
  20     0 moveto    0     0 lineto stroke
```

Le linee, oltre alla collocazione, hanno due caratteristiche importanti: lo spessore e il colore. Lo spessore predefinito dovrebbe essere di un punto, mentre il colore predefinito è il nero. Si modifica lo spessore delle linee con l'istruzione '**setlinewidth**' e la colorazione (grigia) con l'istruzione '**setgray**'. Entrambi ricevono un solo argomento numerico: nel primo caso esprime lo spessore della linea e nel secondo rappresenta la luminosità, con un valore che va da zero a uno (zero rappresenta il nero e uno rappresenta il bianco).

Le istruzioni che alterano le caratteristiche delle linee, hanno effetto solo nel momento in cui appare l'istruzione '**stroke**'. In questo modo, si possono indicare le linee desiderate, quindi si possono cambiare le loro caratteristiche e infine si possono tracciare.

L'esempio seguente disegna lo stesso rettangolo già presentato, specificando un tratto di due punti tipografici di colore grigio (esattamente a metà tra il bianco e il nero). Si può osservare che le istruzioni '**setlinewidth**' e '**setgray**' sono state collocate subito prima dell'istruzione '**stroke**':

```
   0     0 moveto
   0    10 lineto
  20    10 lineto
  20     0 lineto
   0     0 lineto

         2 setlinewidth
         0 setgray
           stroke
```

Si possono tracciare delle linee per disegnare un poligono. L'esempio già visto rappresenta proprio un rettangolo, ma non è stato dichiarato esplicitamente il fatto che le linee devono congiungersi. Per farlo occorre dichiarare un percorso, con l'istruzione '**newpath**', che si conclude con '**closepath**'. Si osservi la variante seguente al disegno del rettangolo:


```
newpath
   0     0 moveto
   0    10 lineto
  20    10 lineto
  20     0 lineto
closepath

   2 setlinewidth
   0 setgray

stroke
```

In pratica, si tracciano le prime tre linee, mentre l'ultima viene indicata implicitamente con la richiesta di chiudere il percorso con l'istruzione '**closepath**'.

Il fatto di avere realizzato un poligono, consente di definire il colore di riempimento. In condizioni normali, quando non è stato fissato alcunché, il poligono è trasparente, mentre se si fissa un riempimento diventa opaco e il colore ricopre anche il bordo tracciato con le linee. Infatti, il bordo ha lo stesso colore fissato con l'istruzione '**setgray**', per cui tutto diventa dello stesso colore. Il colore di riempimento si definisce con l'istruzione '**fill**' e il colore usato è quello già fissato con l'istruzione '**setgray**'.

```
newpath
   0     0 moveto
   0    10 lineto
  20    10 lineto
  20     0 lineto
closepath

% 2   setlinewidth
   0.5 setgray
       fill
stroke
```

L'esempio mostra l'utilizzo dell'istruzione '**fill**' per colorare il rettangolo di grigio. Dal momento che lo spessore delle linee non serve più, l'istruzione relativa è stata commentata. Volendo mettere un bordo a questo rettangolo, occorre ridisegnarne sopra un altro trasparente, con il tratto desiderato:

```
newpath
   0     0 moveto
   0    10 lineto
  20    10 lineto
  20     0 lineto
closepath

   0.5 setgray
       fill
       stroke

newpath
   0     0 moveto
   0    10 lineto
  20    10 lineto
  20     0 lineto
closepath

   2   setlinewidth
   0   setgray
       stroke
```

Per terminare l'argomento sulle linee e sui poligoni, conviene mostrare un esempio completo che poi appare nella figura 48.19. Si osservi che il rettangolo viene disegnato da 1,1 a 21,11, utilizzando l'area da 0,0 a 22,12, la quale costituisce lo stretto indispensabile per dare lo spazio allo spessore della linea che è di due punti.

```
%!PS-Adobe-2.0 EPSF-1.2
%%Creator: Daniele Giacomini
%%BoundingBox: 0 0 22 12
%%EndComments

% Disegna un rettangolo

newpath
   1     1 moveto
   1    11 lineto
```

```
  21    11 lineto
  21     1 lineto
closepath

   0.5 setgray
       fill
       stroke

newpath
   1     1 moveto
   1    11 lineto
  21    11 lineto
  21     1 lineto
closepath

   2   setlinewidth
   0   setgray
       stroke

showpage

%%Trailer
%%EOF
```

Figura 48.19. Il rettangolo riempito e bordato. Il contorno è molto largo, ma è proporzionato rispetto al rettangolo che è alto solo 10 punti.

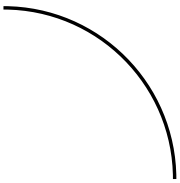
48.2.1 Archi e curve

Si può disegnare un arco o un cerchio completo con l'istruzione '**arc**'. In questo caso, non c'è bisogno di fare riferimento a una posizione corrente; anzi, è meglio eliminare tale informazione con un'istruzione '**stroke**' preventiva. L'istruzione '**arc**' richiede l'indicazione delle coordinate del centro del cerchio, la lunghezza del raggio, l'angolo di partenza e l'angolo di destinazione in direzione antioraria. L'esempio seguente disegna un arco con centro nella posizione ***x*** , ***y***=100 , 150, con raggio di 50 punti, da 0 a 90 gradi:

```
100 150 50 0 90 arc
```

In pratica, si tratta di quanto si vede nella figura 48.21.

Figura 48.21. Arco di cerchio disegnato da 0 a 90 gradi.



Se prima di disegnare il cerchio o l'arco di cerchio si tracciano altre linee, è conveniente chiudere i disegni precedenti con l'istruzione '**stroke**', per evitare di avere delle coordinate correnti attive nel momento in cui si usa l'istruzione '**arc**'. Diversamente, si otterrebbe una linea che collega le coordinate iniziali con il punto di partenza dell'arco disegnato.

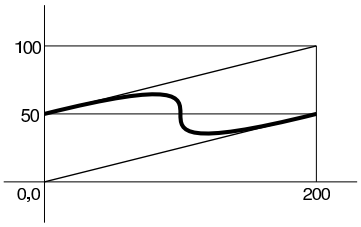
Per disegnare un cerchio completo, basta indicare l'intervallo di angoli da 0 a 360 gradi. Se si vuole riempire il cerchio, non è necessario utilizzare le istruzioni '**newpath**' e '**closepath**', perché si ottiene sempre un riempimento, anche quando il cerchio non è completo.

Il disegno di una curva è invece più complicato: si usa l'istruzione '**curveto**', ma oltre alle coordinate di destinazione, bisogna indicare la tangente del punto di inizio e del punto di destinazione. Prima di dare altre spiegazioni, conviene partire da un esempio visivo, come si vede nella figura 48.23. La curva da prendere in considerazione è rappresentata con un tratto più scuro. Si possono vedere due linee oblique, che partono rispettivamente dal punto di inizio e dal punto di arrivo della curva: si tratta delle tangenti che stabiliscono la

curvatura di partenza e di arrivo della linea disegnata. L'istruzione necessaria a disegnare questa curva è la seguente:

```
    0   50  moveto
  200  100    0    0  200   50  curveto
```

Figura 48.23. Esempio di una curva in cui sono evidenti le tangenti.



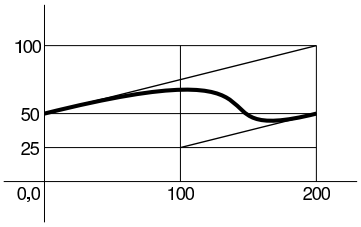
In pratica:

- le coordinate di partenza sono ***x*** , ***y***=0 , 50;
- la prima tangente è la linea che va da ***x*** , ***y***=0 , 50 (le coordinate di partenza) a ***x*** , ***y***=200 , 100;
- la seconda tangente è la linea che va da ***x*** , ***y***=200 , 50 (le coordinate di arrivo) a ***x*** , ***y***=0 , 0;
- le coordinate di arrivo sono ***x*** , ***y***=200 , 50.

Naturalmente, la lunghezza delle linee indicate come tangenti rende più o meno importante la curvatura relativa. Si osservi, nella figura 48.25, come si trasforma il disegno se si accorcia la linea tangente di arrivo come nell'esempio seguente:

```
    0   50  moveto
  200  100    100    25  200   50  curveto
```

Figura 48.25. Esempio di una curva in cui sono evidenti le tangenti.



Il modello sintattico per l'utilizzo dell'istruzione '**curveto**' è quindi il seguente:

```
x_tangente_inizio  y_tangente_inizio  x_tangente_fine  y_tangente_fine ↩
↪x_fine_curva  y_fine_curva  curveto
```

48.3 Salvare e recuperare le impostazioni grafiche

Le impostazioni grafiche, come quelle che si possono fissare con le istruzioni '**setlinewidth**' e '**setgray**', possono essere salvate e recuperate da una pila apposita. Si utilizza l'istruzione '**gsave**' per salvare l'impostazione corrente e '**grestore**' per recuperare le ultime impostazioni salvate. Si osservi l'esempio seguente:

```
1 setlinewidth
gsave
2 setlinewidth
gsave
4 setlinewidth
 50 120  moveto
200 120  lineto
stroke
grestore
 50 125  moveto
200 125  lineto
stroke
grestore
 50 130  moveto
200 130  lineto
stroke
```

Vengono accumulati tre spessori differenti per le linee, quindi si procede disegnando tre linee, dopo ognuna delle quali viene recuperata l'ultima impostazione grafica. In pratica, la linea da 50,120 a 200,120 viene disegnata con un tratto di quattro punti; la linea da 50,125 a 200,125 viene disegnata con un tratto di tre punti; la linea da 50,130 a 200,130 viene disegnata con un tratto di un punto di spessore.

Quando si disegna qualcosa, può essere opportuno racchiudere le modifiche alle caratteristiche entro una coppia '**gsave**'-'**grestore**', onde evitare di coinvolgere le impostazioni precedenti, che potrebbero riguardare il resto del file:

```
gsave
4 setlinewidth
 50 120  moveto
200 120  lineto
stroke
grestore
```

Nell'esempio che appare sopra, si inizia salvando le impostazioni e impostando uno spessore di quattro punti. Viene quindi indicata la linea e fissata con l'istruzione '**stroke**'. Al termine si recuperano le impostazioni grafiche. Nello stesso modo, si potrebbero salvare e modificare le impostazioni grafiche subito prima dell'istruzione '**stroke**':

```
 50 120  moveto
200 120  lineto
gsave
4 setlinewidth
stroke
grestore
```

Attraverso la tecnica del salvataggio e del recupero delle caratteristiche grafiche, è possibile disegnare un poligono bordato senza dover ripetere due volte il tratto del contorno. In pratica, il comando di riempimento viene dato entro un ambiente protetto da '**gsave**' e '**grestore**', come si vede nell'esempio seguente:

```
%!PS-Adobe-2.0 EPSF-1.2
%%Creator: Daniele Giacomini
%%BoundingBox: 0 0 22 12
%%EndComments

% Disegna un rettangolo

newpath
   1     1 moveto
   1    11 lineto
  21    11 lineto
  21     1 lineto
closepath

gsave
   0.5 setgray
       fill
grestore
   2   setlinewidth
   0   setgray
       stroke

showpage

%%Trailer
```



```
%%EOF
```

Il risultato è identico a quanto già visto nella figura 48.19.

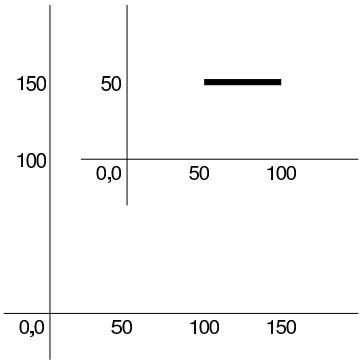
48.4 Spostamento del piano cartesiano e modifica della scala

È possibile spostare la posizione della superficie prima di dare altre istruzioni di scrittura di qualunque tipo. Si tratta in particolare delle istruzioni '**translate**' e di '**rotate**'. La prima di queste due sposta le coordinate 0 , 0 in una posizione nuova, prendendo come riferimento le coordinate precedenti; la seconda ruota gli assi attorno alle coordinate 0 , 0. Per esempio, con le istruzioni seguenti si ottiene una linea orizzontale dalla posizione 100 , 150 alla posizione 150 , 150:

```
 50 100 translate
 50  50 moveto
100  50 lineto
stroke
```

Infatti, l'istruzione '**translate**' sposta le coordinate 0 , 0 verso 50 , 100, secondo la collocazione precedente. La figura 48.31 mostra la collocazione iniziale e lo spostamento; la linea disegnata è quella che appare con tratto più scuro.

Figura 48.31. Esempio riferito a uno spostamento degli assi.

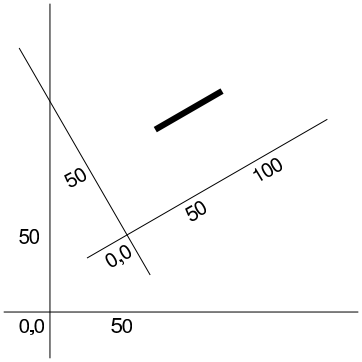


L'istruzione '**rotate**' fa ruotare il piano cartesiano sul centro delle coordinate 0 , 0. La figura 48.33 mostra cosa accade se si sposta lo zero nella posizione 0 , 0 e poi si ruota di 30 gradi, con le istruzioni seguenti:

```
   50   50  translate
   30  rotate

   50   50  moveto
  100   50  lineto
stroke
```

Figura 48.33. Esempio riferito a uno spostamento e a una rotazione degli assi.



Lo spostamento e la rotazione del piano sono informazioni che possono essere salvate e recuperate con le istruzioni '**gsave**' e '**grestore**'. Pertanto, prima di uno spostamento o di una rotazio-

ne, conviene salvare la situazione, per recuperarla quando questi cambiamenti non servono più.

```
gsave
   50   50  translate
   30  rotate

   50   50  moveto
  100   50  lineto
stroke
grestore
```

Oltre allo spostamento e alla rotazione del piano, si può modificare la scala, con l'istruzione '**scale**'. Gli argomenti dell'istruzione sono due valori che esprimono il rapporto nei confronti dell'asse X e nei confronti dell'asse Y. Dopo la modifica della scala, le coordinate 0 , 0 rimangono centrate sulla stessa posizione iniziale. L'esempio seguente serve a fare in modo che la scala dell'asse Y risulti schiacciata alla metà del valore precedente, mentre l'asse X non viene modificato:

```
1 0.5 scale
```

Anche le alterazioni della scala possono essere recuperate da un'istruzione '**grestore**'.

48.5 Ripetizione

Un gruppo di istruzioni, racchiuso tra parentesi graffe, può essere ripetuto più volte con l'istruzione '**repeat**':

```
n {istruzioni} repeat
```

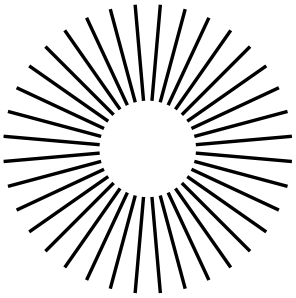
Evidentemente, perché ciò abbia un senso, è necessario che le istruzioni da ripetere creino ogni volta un cambiamento, come una rotazione o uno spostamento di assi. Per esempio,

```
36 {10 rotate 10 10 moveto 30 30 lineto} repeat
stroke
```

Disegna una stellina attorno alle coordinate 0 , 0, con un diametro di 80 punti. In questo caso, la rotazione riporta alla fine gli assi nella posizione di partenza, ma in generale conviene salvare e ripristinare la situazione:

```
gsave
36 {10 rotate 10 10 moveto 30 30 lineto} repeat
stroke
grestore
```

Figura 48.38. Esempio di un disegno ottenuto con la rotazione.



48.6 Testo

Per poter scrivere sulla superficie del foglio, è necessario selezionare il tipo di carattere e la dimensione prima di tutto. Questo si ottiene con alcune istruzioni che conviene usare assieme:

```
/carattere findfont dimensione scalefont setfont
```

Per esempio, per usare il carattere Helvetica alto 12 punti, si usa l'istruzione seguente:

```
/Helvetica findfont 12 scalefont setfont
```

La scrittura vera e propria avviene con l'istruzione '**show**', con la quale si colloca il testo a partire dalle coordinate correnti, per cui si fa precedere normalmente da un'istruzione '**moveto**':


```
(testo) show
```

Come si vede dal modello sintattico, il testo deve essere scritto tra parentesi tonde e collocato prima della parola chiave '**show**'.

Dovendo usare le parentesi tonde per delimitare il testo da visualizzare, deve esserci un modo per poter togliere a queste il valore sintattico normale, quando c'è la necessità di rappresentarle nel testo. In pratica si usano le sequenze '**\(**' e '**\)**' per rappresentarle; inoltre, dal momento che la barra obliqua inversa ha un significato speciale, per rappresentare questa si usa la sequenza '\\'. La tabella 48.40 riepiloga le sequenze speciali più importanti per il testo delimitato tra parentesi tonde.

Tabella 48.40. Alcune sequenze speciali per la scrittura di testo delimitato da parentesi tonde, con l'istruzione '**show**'.

Sequenza	Descrizione
\\	Barra obliqua inversa.
\(	Parentesi tonda aperta.
\)	Parentesi tonda chiusa.
nnn	Simbolo corrispondente al numero, in ottale, secondo la codifica attuale.

Se si seleziona un carattere non disponibile, viene utilizzato il Courier. La tabella 48.41 elenca i nomi standard dei tipi di carattere fondamentali che si possono utilizzare con il linguaggio PostScript.

Tabella 48.41. Nomi dei tipi di carattere comuni che possono essere utilizzati con il linguaggio PostScript.

Nome	Descrizione
Times Times-Roman	Times normale.
Times-Italic	Times corsivo.
Times-Bold	Times neretto.
Times-BoldItalic	Times neretto inclinato.
Helvetica	Helvetica normale.
Helvetica-Oblique	Helvetica inclinato.
Helvetica-Bold	Helvetica neretto.
Helvetica-BoldOblique	Helvetica neretto inclinato.
Courier	Courier normale.
Courier-Oblique	Courier inclinato.
Courier-Bold	Courier neretto.
Courier-BoldOblique	Courier neretto inclinato.
Symbol	Symbol.

Un'istruzione simile a '**show**', si occupa di visualizzare il testo, controllandone lo spostamento in corrispondenza di un simbolo particolare:

```
x y n (testo) widthshow
```

L'istruzione '**widthshow**' serve a fissare uno spostamento orizzontale (***x***) e uno spostamento verticale (***y***), subito dopo il simbolo corrispondente al numero ***n***. In pratica, se alla fine di ogni spazio si vuole aggiungere uno spazio orizzontale di due punti, si usa la forma seguente:

```
2 0 32 (testo) widthshow
```

Questa è anche la situazione tipica, in cui si vuole allargare lo spazio tra le parole, per adattare la riga scritta alla larghezza disponibile. Naturalmente, lo spazio normale può anche essere ridotto, se lo si desidera, utilizzando valori orizzontali negativi, come nell'esempio seguente:

```
-1 0 32 (testo) widthshow
```

Se si attribuisce un valore anche al secondo argomento numerico, si ottiene uno spostamento verticale, come nell'esempio seguente, in cui ogni parola viene alzata di due punti rispetto alla precedente:

```
0 2 32 (testo) widthshow
```

48.7 Salvataggio e recupero delle impostazioni della pagina nel complesso

Quando si scrive un documento composto da diverse pagine, diventa utile la possibilità di recuperare le impostazioni precedenti, prima di passare alla pagina successiva. È già stata presentata la coppia di istruzioni '**gsave**' e '**grestore**', specifica per le impostazioni grafiche. Per tutto, si può usare invece la coppia '**save**' e '**restore**'. Di solito si inizia una pagina con '**save**' e si conclude con '**restore**', in modo da garantire il recupero di tutto, senza dimenticare qualcosa.

```
%!PS-Adobe-2.0
%%Creator: nome_del_redattore_del_file
%%DocumentPaperSizes: formato
%%EndComments
[%%BeginProlog
  prologo
%%EndProlog]
%%Page: numero_mostrato pagina_reale
save
istruzioni_ps
showpage
restore
[%%Page: numero_mostrato pagina_reale
save
istruzioni_ps
showpage
restore]…
%%Trailer
%%EOF
```

Le istruzioni '**save**' e '**restore**' dovrebbero essere sempre annidate correttamente, nel senso che ogni istruzione '**restore**' va a recuperare l'ultima istruzione '**save**' che non sia già stata presa in considerazione da un altro '**restore**'. Se viene eseguito un '**restore**' che non risulta abbinato a un'istruzione '**save**', si genera un errore irreversibile.

Per ovviare all'inconveniente di dover seguire attentamente l'uso delle istruzioni '**save**' e '**restore**', si può attribuire un «nome» a un'istruzione '**save**', richiamando lo stesso nome nel momento del '**restore**':

```
/nome save def
…
…
…
nome restore
```

La cosa non è molto intuitiva, ma funziona così: il nome che viene dichiarato nel momento dell'uso dell'istruzione '**save**', viene posto davanti all'istruzione '**restore**'. Se nello spazio tra queste due istruzioni appaiono altre istruzioni '**save**', queste sono tutte annullate. Segue un esempio:


```
/Mia_Configurazione_Predefinita save def
…
…
…
Mia_Configurazione_Predefinita restore
```

Tabella 48.46. Tabella riassuntiva delle istruzioni più semplici del linguaggio PostScript.

Istruzione	Descrizione
x y moveto	Cambia le coordinate correnti senza disegnare.
x y rmoveto	Cambia le coordinate correnti in modo relativo.
x y lineto	Traccia una linea fino alle coordinate assolute indicate.
x y rlineto	Traccia una linea fino alle coordinate relative indicate.
x y m n o arc	Arco con centro in ***x*** , ***y***, raggio ***m***, da ***n*** a ***o*** gradi, in senso antiorario.
x y m n o arcn	Arco con centro in ***x*** , ***y***, raggio ***m***, da ***n*** a ***o*** gradi, in senso orario.
x y x y x y curveto	Curva indicando le coordinate di arrivo delle tangenti e infine della curva.
x y translate	Fa sì che ***x*** , ***y*** corrispondano alle nuove coordinate 0 , 0.
n rotate	Ruota di ***n*** il fondo, con centro sulle coordinate 0 , 0.
n {*istruzioni*} repeat	Ripete ***n*** volte le istruzioni tra parentesi graffe.
n setlinewidth	Spessore delle linee.
n setgray	Colorazione grigia: 0=nero; 1=bianco.
fill	Riempie i poligoni e le aree racchiuse entro le curve.
newpath	Inizia a disegnare un oggetto nuovo.
closepath	Unisce l'ultimo punto disegnato con il punto di partenza.
stroke	Fissa le linee tracciate.
gsave	Accumula le impostazioni grafiche.
grestore	Recupera le impostazioni.
save	Accumula tutte le impostazioni della pagina.
restore	Recupera le impostazioni della pagina.
/*nome* save def	Accumula tutte le impostazioni dichiarando un nome.
nome restore	Recupera le impostazioni riferite a quel nome.
/*nome* findfont *dim* scalefont setfont	Seleziona il carattere e la dimensione indicata.
(*testo*) show	Scrive il testo indicato usando il carattere già stabilito.
x y n (*testo*) widthshow	Scrive il testo indicato con uno spostamento ***x*** , ***y*** alla fine del simbolo ***n***.
x 0 32 (*testo*) widthshow	Scrive il testo indicato con l'aggiunta di uno spazio di ***x*** tra le parole.

48.8 La pila

Per poter scrivere codice PostScript un po' più complesso, diventa necessario l'utilizzo di istruzioni che realizzano delle espressioni, fino ad arrivare alla costruzione di funzioni (procedure) che possono essere richiamate successivamente. Purtroppo, le espressioni realizzate con questo linguaggio, diventano un po' complicate da leggere. Infatti, queste funzioni ricevono i loro argomenti prelevandoli da una pila (*stack*) ed emettono risultati inserendoli nella stessa pila.

```
dato… funzione
```

Osservando il modello, le informazioni che non sono riconducibili a nomi di funzione, vengono inserite in questa pila, che poi la prima funzione inizia a leggere. Si osservi l'istruzione seguente:

```
1 2 3 4 5 6 7 8 9 moveto lineto
```

I valori da uno a nove, vengono inseriti così come sono nella pila, poi ogni funzione preleva dalla pila la quantità di argomenti che la riguarda. In questo caso, '**moveto**' preleva gli ultimi due valori a essere stati inseriti, precisamente la coppia otto e nove, spostando le coordinate correnti in 8 , 9; successivamente è il turno di '**lineto**', la quale preleva altri due valori, precisamente il sei e il sette, tracciando una linea fino al punto 6 , 7. Pertanto, tutto è come se fosse scritto nel modo seguente:

```
8 9 moveto 6 7 lineto
```

Tuttavia, rimangono ancora altri valori nella pila, per altre funzioni successive, ammesso che vogliano usarli, perché se si inseriscono altri valori, questi ultimi vengono poi estratti per primi.

Dato questo meccanismo, diventano importanti alcune funzioni che consentono di intervenire su questa pila: '**clear**' svuota completamente la pila; '**pop**' preleva ed elimina l'ultimo valore inserito. A fianco di '**pop**' si potrebbe immaginare la presenza di una funzione con il nome *push*, ma in questo caso non serve, perché l'azione di inserimento nella pila avviene in modo implicito.

Sono un po' più difficili da comprendere le funzioni '**exch**' e '**roll**'. La prima scambia l'ordine degli ultimi due valori inseriti nella pila; la seconda esegue uno scorrimento, verso sinistra o verso destra di una certa quantità di questi valori:

```
n_elementi_da_scorrere scorrimento roll
```

Per esempio, se nella pila ci fossero i valori 1, 2, 3, 4, 5, 6 e 7, in questo ordine, per cui il primo a essere prelevato sarebbe il numero 7, l'istruzione '**3 2 roll**' trasformerebbe questa sequenza in 1, 2, 3, 4, 6, 7 e 5; al contrario, l'istruzione '**3 -2 roll**' trasformerebbe questa sequenza in 1, 2, 3, 4, 7, 5 e 6. In pratica, il primo valore indica quanti elementi prendere in considerazione, a partire dall'ultimo, mentre il secondo indica quante volte scorrere e in quale direzione.

Figura 48.49. Esempio di funzionamento dell'istruzione '**roll**', con uno scorrimento verso destra.

```
pila iniziale: 1 2 3 4 5 6 7

    3 2 roll          7 5 6  (primo scorrimento verso destra)
                      6 7 5  (secondo scorrimento verso destra)
pila finale:   1 2 3 4 6 7 5
```

Figura 48.50. Esempio di funzionamento dell'istruzione '**roll**', con uno scorrimento verso sinistra.

```
pila iniziale: 1 2 3 4 5 6 7

    3 -2 roll         6 7 5  (primo scorrimento verso sinistra)
                      7 5 6  (secondo scorrimento verso sinistra)
pila finale:   1 2 3 4 7 5 6
```

Quando una funzione restituisce un valore, lo fa inserendolo implicitamente nella pila. In questo modo, l'assegnamento a una variabile, così come si è abituati nei linguaggi di programmazione comuni, non c'è. Al massimo si definisce una funzione che restituisce un valore, inserendolo nella pila.

Anche le funzioni '**exch**' e '**roll**' prelevano dei valori dalla pila e poi ne inseriscono degli altri nella stessa. In pratica, '**exch**' preleva due valori, li scambia e li inserisce nella pila; '**roll**' preleva due valori, li interpreta, quindi preleva un gruppo di altri valori, li fa scorrere in un verso o nell'altro, quindi li inserisce nuovamente nella pila.

A questo punto si può cominciare a comprendere che i dati inseriti nella pila, quando ciò non avviene per mezzo di una funzione che restituisce qualcosa, devono avere una rappresentazione formale. Può

trattarsi di: valori numerici, che si scrivono come sono, utilizzando il punto per separare la parte decimale; stringhe, che sono delimitate da parentesi tonde e possono contenere delle sequenze di escape; espressioni, che sono delimitate tra parentesi graffe (si ricordi il caso della funzione '**repeat**'). I valori logici, *Vero* e *Falso*, non hanno una rappresentazione particolare e si indicano espressamente solo attraverso le funzioni '**true**' e '**false**'.

Tabella 48.51. Rappresentazione dei dati e gestione della pila.

Istruzione	Descrizione
intero[. *decimale*]	Inserisce il valore numerico nella pila.
(*stringa*)	Inserisce la stringa nella pila.
{ *espressione* }	Inserisce le istruzioni nella pila.
`clear`	Svuota la pila.
oggetto `pop`	Preleva dalla pila l'ultimo valore inserito.
oggetto_1 *oggetto_2* `exch`	Scambia gli ultimi due valori nella pila.
m *n* `roll`	Fa scorrere gli ultimi *m* elementi della pila di *n* posizioni verso destra.
m `-`*n* `roll`	Fa scorrere gli ultimi *m* elementi della pila di *n* posizioni verso sinistra.
oggetto `dup`	Preleva l'ultimo valore e ne inserisce due copie nella pila.

48.9 Funzioni comuni

Alcune funzioni operano su valori numerici, restituendo un risultato che, secondo la logica del linguaggio PostScript, viene inserito nella pila. Per esempio, la funzione '**add**' riceve due valori restituendo la somma di questi:

```
10 20 add 40 moveto
```

In questo caso, vengono sommati i valori 10 e 20, inserendo nella pila il valore 30. Così, si ottiene lo spostamento nelle coordinate 30 , 40, attraverso la funzione '**moveto**'.

I valori logici, come accennato, si indicano attraverso le funzioni '**true**' e '**false**', che si limitano rispettivamente a inserire nella pila il valore corrispondente. Possono generare risultati logici anche alcune funzioni di confronto e i valori logici possono essere rielaborati attraverso funzioni booleane. Infine, in base a un valore logico è possibile eseguire o meno un gruppo di espressioni. Si osservino gli esempi seguenti.

`10 20 lt`	La funzione '**lt**' confronta due valori e restituisce *Vero* se il primo (secondo la lettura umana) è minore. In questo caso, viene restituito *Vero*.
`10 20 lt 45 34 gt and`	La funzione '**and**' restituisce *Vero* se riceve due valori *Vero* simultaneamente. In questo caso, la funzione '**lt**' inserisce il valore *Vero* nella pila e anche la funzione '**gt**' inserisce un altro valore *Vero*, dal momento che il confrontando i valori 45 e 34 si vede che il primo è maggiore del secondo.
`10 20 lt {45 50 moveto} if`	La funzione '**if**' preleva un valore logico e un gruppo di istruzioni. Se il valore logico è *Vero* viene eseguito il raggruppamento di istruzioni. In questo caso, dato che la funzione '**lt**' inserisce il valore *Vero* nella pila, così può essere eseguito lo spostamento nelle coordinate 45 , 50.

`10 20 lt {45 50 moveto} ↩ ↪{50 45 moveto} ifelse`	La funzione '**`ifelse`**' preleva un valore logico e due gruppi di istruzioni. Se il valore logico è *Vero* viene eseguito il primo gruppo di istruzioni, altrimenti viene eseguito il secondo. In questo caso, dato che la funzione '**`lt`**' inserisce il valore *Vero* nella pila, così viene eseguito lo spostamento nelle coordinate 45 , 50.

Queste funzioni vengono descritte brevemente nella tabella 48.54.

Tabella 48.54. Espressioni matematiche, logiche e condizionali.

Istruzione	Descrizione
n `neg`	Inverte il segno del valore.
m ***n*** `add`	Somma i due valori.
m ***n*** `sub`	Sottrae ***n*** da ***m***.
m ***n*** `mul`	Moltiplica i valori.
m ***n*** `div`	Divide ***m*** per ***n***.
m ***n*** `mod`	Il resto della divisione intera di ***m*** per ***n***.
n `round`	Arrotonda ***n***.
n `abs`	Calcola il valore assoluto di ***n***.
n `sin`	Calcola il seno di ***n***.
n `cos`	Calcola il coseno di ***n***.
m ***n*** `min`	Restituisce il minimo tra due valori.
m ***n*** `max`	Restituisce il massimo tra due valori.
`true`	*Vero*.
`false`	*Falso*.
m ***n*** `gt`	*Vero* se ***m*** è maggiore di ***n***.
m ***n*** `ge`	*Vero* se ***m*** è maggiore o uguale a ***n***.
m ***n*** `lt`	*Vero* se ***m*** è minore di ***n***.
m ***n*** `le`	*Vero* se ***m*** è minore o uguale a ***n***.
m ***n*** `eq`	*Vero* se i valori sono uguali.
m ***n*** `ne`	*Vero* se i valori sono diversi.
bool `{`***istruzioni***`} if`	Esegue le istruzioni se il valore logico è *Vero*.
bool `{`***istr_1***`} {`***istr_2***`} ifelse`	Esegue il primo o il secondo gruppo di istruzioni in base al valore logico.

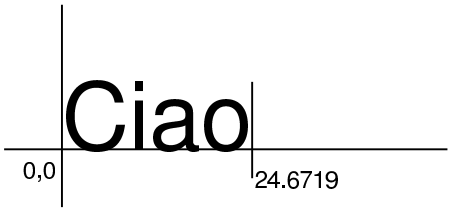
48.10 Operazioni sulle stringhe

A causa della struttura del linguaggio, la gestione delle stringhe non è affatto intuitiva: bisogna tradurre tutto nell'ottica della pila. Tanto per cominciare, la cosa più semplice che si può fare con una stringa è misurarne la lunghezza con l'aiuto della funzione '**`stringwidth`**'. Per la precisione, si tratta di determinare la posizione finale di una stringa collocata a partire dalle coordinate 0 , 0:

```
stringa stringwidth
```

Se si osserva la figura 48.55, si può vedere la stringa composta dalla parola «Ciao», scritta con il carattere Helvetica, avente un corpo di 12 punti. Come si vede, la sua lunghezza è di 24,672 punti.

Figura 48.55. Lunghezza di una stringa.



Quando c'è la necessità di convertire un valore in una stringa, si pone il problema dell'allocazione di memoria per la stringa stessa. Per esempio, la funzione '**`cvs`**' converte un valore in stringa, ma per farlo deve avere già una stringa da prelevare dalla pila:

```
valore stringa cvs
```

Volendo convertire il valore 23,45 in stringa, bisogna preparare prima una stringa di almeno cinque caratteri:

```
23.45 (     ) cvs
```

Per allocare una stringa, composta da caratteri <*NUL*>, ovvero 000_8, si può usare la funzione '**`string`**', che richiede l'indicazione della quantità di caratteri. Pertanto, la stessa cosa avrebbe potuto essere scritta nel modo seguente:

```
23.45 5 string cvs
```

Naturalmente, la funzione '**`cvs`**' si può usare per visualizzare la stringa generata, per esempio nel modo seguente:

```
10 10 moveto
23.45
5 string cvs
show
```

Si osservi che con '**`cvs`**', anche se si alloca una stringa più grande del necessario, questa viene ridotta alla dimensione richiesta dalla conversione.

Tabella 48.59. Espressioni relative a stringhe.

Istruzione	Descrizione
n `string`	Alloca una stringa di ***n*** caratteri <*NUL*>.
stringa `stringwidth`	Inserisce le coordinate finali della stringa nella pila.
valore ***stringa*** `cvs`	Restituisce una stringa corrispondente al valore.
valore ***n*** `string cvs`	Restituisce una stringa corrispondente al valore, con un massimo di ***n*** caratteri.

48.11 Funzioni

Una funzione si definisce attraverso la sintassi seguente:

```
/nome {istruzioni} def
```

In pratica, si vuole fare in modo che usando il nome indicato, si faccia riferimento automaticamente al gruppo di istruzioni contenuto tra parentesi graffe. Si noti che, eccezionalmente, se si tratta di definire una costante o se si vuole ridefinire il nome di un'altra funzione, non sono necessarie le parentesi graffe.

Come per qualunque altra funzione normale, anche le funzioni definite in questo modo ricevono gli argomenti delle chiamate dalla pila. Per esempio, la funzione '**`quadrilatero`**' che si potrebbe dichiarare nel modo seguente, va usata mettendo davanti, ordinatamente gli argomenti per le varie funzioni utilizzate:

```
/quadrilatero { newpath moveto lineto lineto lineto ↩
↪closepath stroke } def
```

Per esempio, volendo disegnare un quadrato con gli angoli nelle coordinate 0 , 0, 0 , 10, 10 , 10 e 10 , 0, si deve usare la funzione '**`quadrilatero`**' nel modo seguente:

```
10 0 10 10 0 10 0 0 quadrilatero
```

È importante osservare che la prima coppia di coordinate è quella presa in considerazione dall'ultima funzione '**lineto**' contenuta nel raggruppamento di '**quadrilatero**'.

Così come si definisce una funzione, si può attribuire a un nome un valore costante. In questi casi eccezionali, è consentita l'eliminazione delle parentesi graffe:

```
/nome costante def
```

Con la definizione di costanti, si può stabilire una volta per tutte il valore di qualcosa, come nell'esempio seguente:

```
/Margine_Sinistro 80 def
/Margine_Destro 80 def
/Margine_Superiore 100 def
/Margine_Inferiore 100 def
```

48.11.1 Variabili

Nel linguaggio PostScript non è prevista la gestione di variabili: tutto viene elaborato attraverso la pila. Tuttavia, esiste un trucco per ottenere qualcosa che assomigli a delle variabili; si tratta di sfruttare opportunamente la definizione di funzioni. È già stato visto l'assegnamento di un valore costante a un nome:

```
/nome costante def
```

Oppure:

```
/nome { costante } def
```

Se si vuole attribuire a una funzione un valore diverso, occorre un trucco, che si può schematizzare come segue:

```
/nome_1 { /nome_2 exch def } def
```

Si tratta di una funzione che ne dichiara un'altra, ma si osservi con attenzione: la parola chiave '**exch**' non è racchiusa tra parentesi graffe e non può esserlo, se si vuole che il meccanismo funzioni.

Per assegnare un valore alla funzione ***nome_2***, si utilizza una chiamata alla funzione ***nome_1***:

```
n nome_1
```

Per leggere il valore, si fa riferimento alla funzione ***nome_2***, come nell'esempio seguente in cui si utilizza questo dato come coordinata Y per uno spostamento:

```
n nome_1
m nome_2 moveto
```

48.12 Dizionari

La dichiarazione delle funzioni può essere inserita in un dizionario, da richiamare quando serve e da sostituire eventualmente con altre di un altro dizionario, quando la situazione lo richiede. In pratica, la definizione di dizionari di funzioni consente di fare riferimento a gruppi di funzioni solo nell'ambito di un certo contesto, ripristinando un utilizzo differente delle stesse subito dopo.

```
/dizionario n dict def
dizionario begin
    dichiarazione_di_funzione
    …
    …
end
```

Il modello sintattico mostra in che modo procedere alla dichiarazione di un dizionario. Si può osservare che prima della parola chiave '**dict**' occorre indicare un numero, allo scopo di definire una quantità di memoria da allocare per il dizionario stesso. Per abilitare l'uso delle funzioni dichiarate nel dizionario, si deve dichiarare espressamente:

```
dizionario begin
    istruzione
    …
    …
end
```

Eventualmente, la dichiarazione di utilizzo di un dizionario si può annidare; quando si raggiunge la parola chiave '**end**', termina il campo di azione dell'ultimo dizionario ancora aperto.

In generale, potrebbe essere conveniente inserire la dichiarazione dei dizionari nell'ambito delle istruzioni speciali '**%%BeginProlog**' e '**%%EndProlog**'. L'esempio seguente mostra un estratto di un file PostScript ipotetico, in cui si dichiara un dizionario (molto breve) e lo si utilizza immediatamente nell'ambito della pagina (i puntini di sospensione indicano una parte mancante del file che non viene mostrata).

```
%!PS-Adobe-2.0
%%DocumentPaperSizes: a4
%%EndComments
%%BeginProlog
/Mio_Dizionario 50 dict def
Mio_Dizionario begin
    /quadrilatero
    {
        newpath moveto lineto lineto lineto closepath stroke
    } def
    /Margine_Sinistro 80 def
    /Margine_Destro 80 def
    /Margine_Superiore 100 def
    /Margine_Inferiore 100 def
end     % Fine della dichiarazione del dizionario
        % «Mio_Dizionario»
%%EndProlog
Mio_Dizionario begin
%%Page: 1 1
…
…
…
showpage
%%Page: 2 2
…
…
…
showpage
end     % Fine del campo di azione del dizionario
        % «Mio_Dizionario»
%%Trailer
%%EOF
```

48.13 Caratteri da stampa

Il linguaggio PostScript, nelle sue prime versioni, prevede che il file contenente le istruzioni sia di tipo ASCII, dove è ammissibile usare gli 8 bit per esteso. Tuttavia, l'insieme di caratteri a disposizione non è il solito ISO 8859-1, ma qualcosa di diverso che può essere visto nella tabella 48.64.

Tabella 48.64. Insieme di caratteri normale del linguaggio PostScript.

	*															
\000																\017
\020																\037
\040		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/	\057
\160	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?	\077
\100	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	\117
\120	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_	\137
\140	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	\157
\160	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~		\177
\200																	\217
\220																	\237
\240		¡	¢	£	⁄	¥	ƒ	§	¤	'	"	«	‹	›	fi	fl	\257
\260		–	†	‡	·		¶	•	‚	„	”	»	…	‰		¿	\277
\300		`	´	ˆ	˜	¯	˘	˙	¨		°	¸		˝	˛	ˇ	\317
\320	—																\337
\340		Æ		ª					Ł	Ø	Œ	º					\357
\360		æ				ı			ł	ø	œ	ß					\377

Per fare riferimento a un simbolo in forma numerica, basta indicare il numero ottale corrispondente, di tre cifre, preceduto dalla barra obliqua inversa. Per facilitare la lettura, la tabella mostra questo modo di fare riferimento ai caratteri, all'inizio e alla fine di ogni riga.

Per poter usare la codifica corrispondente allo standard ISO 8859-1, è necessario dare delle istruzioni particolari. Volendo sbirciare nel codice PostScript generato da vari sistemi di composizione, si può osservare che ognuno utilizza un proprio modo, più o meno sofisticato. Probabilmente, la tecnica più semplice è quella che si può leggere nei sorgenti di Ghostscript, dove vengono dichiarate due funzioni apposite nel modo seguente:

```
/font-to-iso-latin-1
{
    dup length dict begin {1 index /FID ne {def} {pop pop}
    ifelse} forall
    /Encoding ISOLatin1Encoding def currentdict end
    dup /FontName get 80 string cvs (-ISOLatin1)
    concatstrings cvn
    exch definefont
} def

/find-latin-font
{
  findfont font-to-iso-latin-1
} def
```

In questo modo, quando si va a dichiarare l'uso di un certo tipo di carattere da stampa, invece dell'istruzione seguente, si usa piuttosto quella successiva:

```
/carattere findfont dimensione scalefont setfont
```

Forma modificata per la selezione della codifica ISO 8859-1:

```
/carattere find-latin-font dimensione scalefont setfont
```

48.14 Aspetto dei caratteri da stampa comuni

Il linguaggio PostScript mette a disposizione pochi tipi di carattere da stampa; tuttavia, è disponibile l'interprete Ghostscript, su moltissime piattaforme, che mette a disposizione un numero discreto di questi tipi. La tabella 48.76 elenca brevemente i nomi di questi tipi di carattere, mentre in altre tabelle successive viene mostrato l'aspetto di alcuni di questi, ordinando i simboli secondo l'insieme ISO 8859-1.

Tabella 48.66. Carattere da stampa AvantGarde.

	*															
\000																\017
\020																\037
\040		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/	\057
\160	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?	\077
\100	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	\117
\120	P	Q	R	S	T	U	V	W	X	Y	Z	(	\	)	^	_	\137
\140	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	\157
\160	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~		\177
\200																	\217
\220	ı	`	´	ˆ	˜	¯	˘	˙	¨		°	¸		˝	˛	ˇ	\237
\240		¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	-	®	¯	\257
\260	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿	\277
\300	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï	\317
\320	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß	\337
\340	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï	\357
\360	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ	\377

Tabella 48.67. Carattere da stampa Bookman.

	*															
\000																\017
\020																\037
\040		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/	\057
\160	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?	\077
\100	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	\117
\120	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_	\137
\140	‘	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	\157
\160	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~		\177
\200																	\217
\220	ı	`	´	ˆ	˜	¯	˘	·	¨		°	¸		″	˛	ˇ	\237
\240		¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	-	®	¯	\257
\260	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿	\277
\300	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï	\317
\320	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß	\337
\340	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï	\357
\360	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ	\377

Tabella 48.68. Carattere da stampa Courier.

	*															
\000																\017
\020																\037
\040		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/	\057
\160	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?	\077
\100	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	\117
\120	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_	\137
\140	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	\157
\160	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~		\177
\200																	\217
\220	ı	`	´	ˆ	˜	¯	˘	·	¨		°	¸		″	˛	ˇ	\237
\240		¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	-	®	¯	\257
\260	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿	\277
\300	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï	\317
\320	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß	\337
\340	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï	\357
\360	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ	\377

Tabella 48.69. Carattere da stampa Helvetica.

	*															
\000																\017
\020																\037
\040		!	"	#	$	%	&	’	(	)	*	+	,	–	.	/	\057
\160	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?	\077
\100	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	\117
\120	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_	\137
\140	‘	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	\157
\160	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~		\177
\200																	\217
\220	ı	`	´	ˆ	˜	¯	˘	˙	¨		°	¸		˝	˛	ˇ	\237
\240		¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	-	®	¯	\257
\260	°	±	2	3	´	µ	¶	·	¸	1	º	»	¼	½	¾	¿	\277
\300	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï	\317
\320	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß	\337
\340	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï	\357
\360	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ	\377

Tabella 48.70. Carattere da stampa NewCenturySchlbk.

	*															
\000																\017
\020																\037
\040		!	"	#	$	%	&	’	(	)	*	+	,	–	.	/	\057
\160	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?	\077
\100	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	\117
\120	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_	\137
\140	‘	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	\157
\160	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~		\177
\200																	\217
\220	ı	`	´	ˆ	˜	¯	˘	˙	¨		°	¸		˝	˛	ˇ	\237
\240		¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	-	®	¯	\257
\260	°	±	2	3	´	µ	¶	·	¸	1	º	»	¼	½	¾	¿	\277
\300	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï	\317
\320	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß	\337
\340	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï	\357
\360	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ	\377

Tabella 48.71. Carattere da stampa Palatino.

	*															
\000																\017
\020																\037
\040		!	"	#	$	%	&	’	(	)	*	+	,	–	.	/	\057
\160	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?	\077
\100	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	\117
\120	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_	\137
\140	‘	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	\157
\160	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~		\177
\200																	\217
\220	ı	`	´	ˆ	˜	¯	˘	˙	¨		°	¸		˝	˛	ˇ	\237
\240		¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	-	®	¯	\257
\260	°	±	2	3	´	µ	¶	·	¸	1	º	»	¼	½	¾	¿	\277
\300	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï	\317
\320	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß	\337
\340	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï	\357
\360	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ	\377

Tabella 48.72. Carattere da stampa Times.

	*															
\000																\017
\020																\037
\040		!	"	#	$	%	&	’	(	)	*	+	,	–	.	/	\057
\160	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?	\077
\100	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	\117
\120	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_	\137
\140	‘	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	\157
\160	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~		\177
\200																	\217
\220	ı	`	´	ˆ	˜	¯	˘	˙	¨		°	¸		˝	˛	ˇ	\237
\240		¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	-	®	¯	\257
\260	°	±	2	3	´	µ	¶	·	¸	1	º	»	¼	½	¾	¿	\277
\300	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï	\317
\320	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß	\337
\340	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï	\357
\360	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ	\377

Tabella 48.73. Carattere da stampa ZapfChancery.

	*															
\000																\017
\020																\037
\040		!	"	#	$	%	&	’	(	)	*	+	,	–	.	/	\057
\160	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?	\077
\100	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	\117
\120	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_	\137
\140	‘	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	\157
\160	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~		\177
\200																	\217
\220	ı	`	´	ˆ	˜	¯	˘	˙	¨		°	¸		˝	˛	ˇ	\237
\240		¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	-	®	¯	\257
\260	°	±	2	3	´	µ	¶	·	¸	1	º	»	¼	½	¾	¿	\277
\300	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï	\317
\320	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß	\337
\340	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï	\357
\360	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ	\377

Tabella 48.74. Carattere da stampa ZapfDingbats (solo con l’insieme di caratteri normale).

	*															
\000																\017
\020																\037
\040		✁	✂	✃	✄	☎	✆	✇	✈	✉	☛	☞	✌	✍	✎	✏	\057
\160	✐	✑	✒	✓	✔	✕	✖	✗	✘	✙	✚	✛	✜	✝	✞	✟	\077
\100	✠	✡	✢	✣	✤	✥	✦	✧	★	✩	✪	✫	✬	✭	✮	✯	\117
\120	✰	✱	✲	✳	✴	✵	✶	✷	✸	✹	✺	✻	✼	✽	✾	✿	\137
\140	❀	❁	❂	❃	❄	❅	❆	❇	❈	❉	❊	❋	●	❍	■	❑	\157
\160	❒	❑	❒	▲	▼	◆	❖	◗	|	❙	❚	❛	❜	❝	❞		\177
\200																	\217
\220																	\237
\240		❡	❢	❣	❤	❥	❦	❧	♣	♦	♥	♠	①	②	③	④	\257
\260	⑤	⑥	⑦	⑧	⑨	⑩	❶	❷	❸	❹	❺	❻	❼	❽	❾	❿	\277
\300	①	②	③	④	⑤	⑥	⑦	⑧	⑨	⑩	❶	❷	❸	❹	❺	❻	\317
\320	❼	❽	❾	❿	➔	→	↔	↕	➘	➙	➚	➛	➜	→	➝	➞	\337
\340	➟	➠	➢	➢	➤	➥	➦	➧	➨	➩	➪	➫	➬	➭	➮	➯	\357
\360		➱	➲	➳	➴	➵	➶	➷	➸	➹	➺	➻	➼	➽	➾		\377

Tabella 48.75. Carattere da stampa Symbol (solo con l'insieme di caratteri normale).

\000																\017
\020																\037
\040		!	∀	#	∃	%	&	∍	(	)	*	+	,	−	.	/	\057
\160	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?	\077
\100	≅	Α	Β	Χ	Δ	Ε	Φ	Γ	Η	Ι	ϑ	Κ	Λ	Μ	Ν	Ο	\117
\120	Π	Θ	Ρ	Σ	Τ	Υ	ς	Ω	Ξ	Ψ	Ζ	[	∴	]	⊥	_	\137
\140	‾	α	β	χ	δ	ε	φ	γ	η	ι	φ	κ	λ	μ	ν	ο	\157
\160	π	θ	ρ	σ	τ	υ	ϖ	ω	ξ	ψ	ζ	{	|	}	~		\177
\200																	\217
\220																	\237
\240	∈	ϒ	′	≤	⁄	∞	ƒ	♣	♦	♥	♠	↔	←	↑	→	↓	\257
\260	°	±	″	≥	×	∝	∂	•	÷	≠	≡	≈	…	|	—	↵	\277
\300	ℵ	ℑ	ℜ	℘	⊗	⊕	∅	∩	∪	⊃	⊇	⊄	⊂	⊆	∈	∉	\317
\320	∠	∇	®	©	™	∏	√	·	¬	∧	∨	⇔	⇐	⇑	⇒	⇓	\337
\340	◊	〈	®	©	™	∑	⎛	⎜	⎝	⎡	⎢	⎣	⎧	⎨	⎩	⎪	\357
\360		〉	∫	⌠	⎮	⌡	⎞	⎟	⎠	⎤	⎥	⎦	⎫	⎬	⎭		\377

Tabella 48.76. Nomi dei tipi di carattere da stampa che solitamente sono disponibili con Ghostscript.

Nome	Descrizione
AvantGarde-Book	AvantGarde normale.
AvantGarde-BookOblique	AvantGarde inclinato.
AvantGarde-Demi	AvantGarde neretto.
AvantGarde-DemiOblique	AvantGarde neretto obliquo.
Bookman-Demi	Bookman neretto.
Bookman-DemiItalic	Bookman neretto corsivo.
Bookman-Light	Bookman normale.
Bookman-LightItalic	Bookman corsivo.
Courier	Courier normale.
Courier-Bold	Courier neretto.
Courier-BoldOblique	Courier neretto inclinato.
Courier-Oblique	Courier inclinato.
Helvetica	Helvetica normale.
Helvetica-Bold	Helvetica neretto.
Helvetica-BoldOblique	Helvetica neretto inclinato.
Helvetica-Narrow	Helvetica stretto.
Helvetica-Narrow-Bold	Helvetica stretto neretto.
Helvetica-Narrow-BoldOblique	Helvetica stretto neretto inclinato.
Helvetica-Narrow-Oblique	Helvetica stretto inclinato.
Helvetica-Oblique	Helvetica inclinato.
NewCenturySchlbk-BoldItalic	NewCenturySchlbk neretto corsivo.
NewCenturySchlbk-Bold	NewCenturySchlbk neretto.
NewCenturySchlbk-Italic	NewCenturySchlbk corsivo.
NewCenturySchlbk-Roman	NewCenturySchlbk normale.
Palatino-Bold	Palatino neretto.
Palatino-BoldItalic	Palatino neretto corsivo.
Palatino-Italic	Palatino corsivo.
Palatino-Roman	Palatino normale.
Times-Bold	Times neretto.
Times-BoldItalic	Times neretto inclinato.
Times-Italic	Times corsivo.
Times-Roman	Times normale.
ZapfChancery-MediumItalic	ZapfChancery-MediumItalic medio corsivo.
ZapfDingbats	ZapfDingbats (solo con l'insieme di caratteri normale).
Symbol	Symbol (solo con l'insieme di caratteri normale).

48.15 Distorsione e spostamento dei caratteri da stampa

Fino a questo punto è stato visto come selezionare un carattere da stampa con l'istruzione seguente o con quella successiva, nel caso sia disponibile la funzione relativa:

```
/carattere findfont dimensione scalefont setfont
```

```
/carattere find-latin-font dimensione scalefont setfont
```

In alternativa, è possibile sostituire l'indicazione della dimensione con qualcosa di più articolato, secondo uno dei due schemi seguenti:

```
/carattere findfont [dim_x rot_x rot_y dim_y sp_x sp_y] ↩
↪makefont setfont
```

```
/carattere find-latin-font [dim_x rot_x rot_y dim_y sp_x sp_y] ↩
↪makefont setfont
```

Tra parentesi quadre appaiono dei valori che, se non utilizzati, vanno lasciati azzerati. Per comprenderne il significato, conviene partire dalla situazione normale. Supponendo di volere dichiarare un carattere da stampa di tipo Helvetica, normale, di 12 punti, si può usare la definizione solita:

```
/Helvetica findfont 12 scalefont setfont
```

Oppure, in modo più complesso, quella seguente:

```
/Helvetica findfont [12 0 0 12 0 0] makefont setfont
```

In pratica, ***dim_x*** e ***dim_y*** servono per definire la dimensione orizzontale e verticale del carattere. In condizioni normali, la dimensione è la stessa; diversamente, con una dimensione orizzontale più grande di quella verticale si ottiene un carattere più largo, mentre con una dimensione verticale maggiore di quella orizzontale, si ottiene un carattere alto (o sottile). Si osservi l'esempio seguente, di un carattere Helvetica allargato e successivamente alzato. Il risultato, in rapporto, si può vedere nella figura 48.81.

```
/Helvetica findfont [30 0 0 12 0 0] makefont setfont
```

```
/Helvetica findfont [12 0 0 30 0 0] makefont setfont
```

Figura 48.81. Distorsione orizzontale e verticale del carattere da stampa.

Helvetica allargato

Helvetica alzato (stretto)

I valori corrispondenti a ***rot_x*** e ***rot_y*** permettono di inclinare l'asse X o l'asse Y del testo. In pratica, modificando ***rot_x*** si cambia l'andamento orizzontale del testo secondo l'angolo indicato. Per esempio un angolo di 10 gradi fa sì che la riga scritta sia inclinata verso l'alto (ovvero di 10 gradi in senso antiorario), mentre un angolo di -10 gradi genera una riga inclinata verso il basso. La figura 48.84 mostra il risultato che si può ottenere con le due distorsioni seguenti:

```
/Helvetica findfont [12 5 0 12 0 0] makefont setfont
```

```
/Helvetica findfont [12 -5 0 12 0 0] makefont setfont
```

Figura 48.84. Distorsione dell'inclinazione orizzontale.

Scrittura ascendente

Scrittura discendente

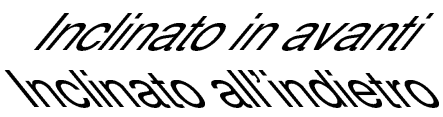
Intervenendo invece nel valore ***rot_y***, si cambia l'inclinazione del carattere, senza cambiare l'andamento della riga. In pratica, valori

positivi dell'angolazione generano un'inclinazione in avanti, simile a un corsivo, mentre valori negativi generano un'inclinazione all'indietro. La figura 48.87 mostra il risultato che si può ottenere con le due distorsioni seguenti:

```
/Helvetica findfont [12 0 10 12 0 0] makefont setfont
```

```
/Helvetica findfont [12 0 -10 12 0 0] makefont setfont
```

Figura 48.87. Distorsione dell'inclinazione verticale.



I valori corrispondenti a ***sp_x*** e ***sp_y*** rappresentano uno spostamento orizzontale e verticale, in punti, senza applicare delle distorsioni vere e proprie.

L'andamento del testo, che normalmente si svolge da sinistra a destra, può essere invertito, assegnando un valore negativo per il primo valore, ovvero per ***dim_x***. La figura 48.89 mostra la comparazione tra un testo scritto in modo normale e un altro che invece si sviluppa verso sinistra, con un comando simile a quello seguente:

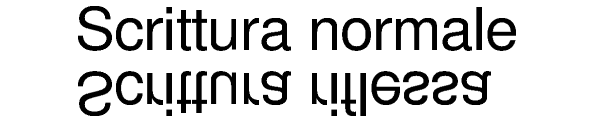
```
/Helvetica findfont [-12 0 0 12 0 0] makefont setfont
```

Figura 48.89. Scrittura normale e scrittura da destra a sinistra.

Così come si può invertire l'andamento del testo in orizzontale, si può invertire in modo verticale, ottenendo una sorta di riflessione. La figure 48.91 mette a confronto un testo scritto in modo normale e un altro modificato con l'istruzione seguente:

```
/Helvetica findfont [12 0 0 -12 0 0] makefont setfont
```

Figura 48.91. Scrittura ruotata sull'asse orizzontale.



48.16 Esempi di funzioni

In questa sezione si raccolgono alcuni esempi di funzioni PostScript che possono essere utili a vario titolo. Naturalmente, trattandosi di un linguaggio specifico per la stampa, non vengono proposti esempi di programmazione standard.

48.16.1 Unità di misura

L'unità di misura utilizzata è sempre il punto tipografico, che in questo ambito corrisponde a 1/72-esimo di pollice, ovvero a 0,3527777 mm (per converso, un millimetro è pari a 2,834646 punti). Volendo usare unità di misura più comuni, si possono realizzare alcune funzioni molto semplici che si limitano a moltiplicare il valore per una costante, in modo da ottenere come risultato l'equivalente in punti:

```
/cm { 28.346456 mul } def  % <n> cm % converte 'cm' in punti
/mm { 2.8346456 mul } def  % <n> mm % converte 'mm' in punti
/pt { 1 mul } def          % <n> pt % non converte alcunché
/in { 72 mul } def         % <n> in % converte 'in' in punti
```

In questo modo, al posto di inserire un valore puro e semplice, basta aggiungere subito dopo la sigla dell'unità di misura, che in realtà è una funzione. Per esempio:

```
3 cm 18 cm moveto
```

Evidentemente, la funzione '**pt**' è inutile, ma può servire per mantenere coerenza con il resto, nel momento in cui si utilizzi sistematicamente questo meccanismo per indicare le coordinate o le distanze.

48.16.2 Funzioni diagnostiche

Pur non trattandosi di un linguaggio di programmazione normale, quando si cerca di realizzare qualcosa di particolare, può essere comoda la possibilità di mostrare un valore da qualche parte, per verificare il contenuto di una data informazione.

```
/diag_display_number % <n> <x> <y> diag_display_number
{
    gsave
    /Helvetica findfont 10 scalefont setfont
    moveto
    100 string cvs show
    grestore
} def
```

La funzione appena mostrata, serve per ottenere la conversione di un numero in stringa, che poi viene visualizzato nelle coordinate previste. Andrebbe usata nel modo seguente, dove ***x*** , ***y*** sono le coordinate a partire dalle quali mostrare il valore:

```
n x y diag_display_number
```

Dal momento che questa funzione preleva il valore dalla pila, potrebbe essere conveniente la duplicazione di tale valore prima di utilizzarlo:

```
n dup x y diag_display_number
```

Volendo completare il problema con una funzione equivalente per la visualizzazione delle stringhe, basta la variante seguente:

```
/diag_display_string % <stringa> <x> <y> diag_display_string
{
    gsave
    /Helvetica findfont 10 scalefont setfont
    moveto
    show
    grestore
} def
```

48.16.3 Gestione di stringhe

Quando si disegnano delle figure o dei grafici, può essere comodo disporre di qualche funzione che faciliti la collocazione di didascalie o di annotazioni di qualunque tipo. Qui viene proposto un sistema molto semplice, con cui è possibile piazzare delle stringhe allineate correttamente a sinistra, a destra o al centro, date le coordinate di riferimento. Si parte con la definizione di alcune «variabili», che servono per fissare i punti di riferimento delle stringhe:

```
/set_line_t   { /line_t    exch def } def
/set_line_l   { /line_l    exch def } def
/set_line_r   { /line_r    exch def } def
/set_line_c   { /line_c    exch def } def
/set_line_h   { /line_h    exch def } def
```

La funzione '**line_t**' viene usata per conoscere la posizione verticale (Y) della stringa da collocare; la funzione '**line_l**' serve per fornire la posizione iniziale (X) di una stringa da allineare a sinistra; '**line_r**' fornisce la posizione finale (X) di una stringa da allineare a destra; '**line_c**' fornisce la posizione centrale (X) di una stringa da centrare; infine, '**line_h**' serve per conoscere la distanza tra le righe.

Per usare le funzioni che vengono presentate nel seguito, devono essere impostati inizialmente i valori per le variabili appena descritte. Per esempio, sapendo che si vuole scrivere un testo allineato a sinistra a partire dalla coordinata 100 , 50, con una distanza tra le righe di 14 punti, basta impostare i valori nel modo seguente:

```
 50 set_line_t
100 set_line_l
 14 set_line_h
```

Dal momento che non si devono centrare le righe e nemmeno allineare a destra, le altre variabili non servono, altrimenti occorrerebbe impostare '**line_c**' o '**line_r**', al posto di '**line_l**'.

```
/show_line_left % <stringa> show_line_left
{
    % Si posiziona all'inizio della riga.
    line_l line_t moveto

    % Mostra la riga.
    show

    % Sottrae alla posizione Y l'altezza della riga.
    line_t line_h sub set_line_t
} def
```

Si osservi la funzione '**show_line_left**': si usano i valori restituiti dalle funzioni '**line_l**' e '**line_t**' per impostare le coordinate iniziali, quindi si visualizza la stringa. Subito dopo si provvede a ridurre il valore della variabile corrispondente alla funzione '**line_t**' del valore restituito da '**line_h**' (la distanza tra le righe), in modo da poter continuare con la visualizzazione di altre stringhe, con lo stesso allineamento sinistro, subito sotto quella appena inserita.

```
/show_line_right % <stringa> show_line_right
{
    % Calcola la lunghezza della riga.
    dup % fa una copia della stringa
    stringwidth % restituisce le coordinate finali X e Y
    pop % elimina la coordinata Y

    % Sottrae dalla posizione destra la lunghezza della
    % riga, cambiando poi il segno.
    line_r sub  neg

    % Si posiziona correttamente.
    line_t moveto

    % Mostra la riga e va all'inizio di una riga nuova.
    show

    % Sottrae alla posizione Y l'altezza della riga.
    line_t line_h sub set_line_t
} def
```

La funzione '**show_line_right**' è molto simile, con la differenza che occorre fare qualche calcolo per individuare la posizione orizzontale di inizio, sapendo la posizione finale ottenuta dalla funzione '**line_r**'. Per questo, viene fatta una copia della stringa, che quindi viene misurata con la funzione '**setlinewidth**'. Da questa misurazione si espelle l'informazione verticale, che risulta inutile, sottraendo poi a questa il valore restituito da '**line_r**'. Quello che si ottiene è la distanza dalla posizione destra finale, con segno invertito, pertanto si inverte nuovamente con la funzione '**neg**'. Disponendo della coordinata X, si aggiunge la coordinata Y, data da '**line_t**', spostando la posizione corrente, per poi mostrare la stringa e infine preparare nuovamente la posizione verticale per una nuova riga.

```
/show_line_center
{
    % Calcola la lunghezza della riga.
    dup % fa una copia della stringa
    stringwidth % restituisce le coordinate finali X e Y
    pop % elimina la coordinata Y
    2 div % divide la lunghezza a metà

    % Sottrae dal centro la metà della riga, cambiando poi
    % il segno.
    line_c sub  neg

    % Si posiziona correttamente.
    line_t moveto

    % Mostra la riga e va all'inizio di una riga nuova.
    show

    % Sottrae alla posizione Y l'altezza della riga.
    line_t line_h sub set_line_t
} def
```

La funzione '**show_line_center**' centra la stringa in riferimento alla posizione '**line_c**'. I calcoli sono simili a quelli per l'allineamento a destra, con la differenza che la distanza dal centro è pari alla metà della lunghezza della stringa. Il resto è equivalente.

L'esempio seguente mostra come scrivere qualcosa con queste funzioni; la figura 48.102 mostra il risultato che si ottiene osservando il riquadro che va da 0 , 0 a 150 , 100:

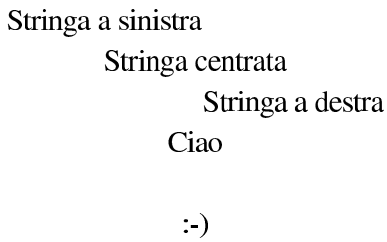
```
/Times-Roman findfont 10 scalefont setfont

 90 set_line_t
 10 set_line_l
140 set_line_r
 75 set_line_c
 14 set_line_h

(Stringa a sinistra) show_line_left
(Stringa centrata) show_line_center
(Stringa a destra) show_line_right
(Ciao) show_line_center
() show_line_center
(:-\)) show_line_center

showpage
```

Figura 48.102. Esempio nell'uso delle funzioni sulle stringhe.



Naturalmente, si possono predisporre anche delle abbreviazioni a queste funzioni:

```
/L { show_line_left } def
/R { show_line_right } def
/C { show_line_center } def
```

In questo modo, l'indicazione delle stringhe può essere ridotto alla forma seguente:

```
(Stringa a sinistra) L
(Stringa centrata) C
(Stringa a destra) R
(Ciao) C
() C
(:-\)) C
```

48.17 Approfondimento: modifica sistematica dei contenuti

Un problema abbastanza frequente negli uffici è la creazione di un'immagine da sovrapporre a tutte le stampate, come un marchio, o come una fincatura. Per esempio, al posto di far stampare della carta intestata, si vuole fare in modo che la stampante stampi automaticamente l'intestazione, senza fare la cosa in due passaggi. Questa sezione cerca di mostrare un metodo pratico per risolvere il problema.

48.17.1 Modifica del file PostScript

Per aggiungere qualcosa a una stampa si deve intervenire nel file PostScript prima che questo raggiunga la coda di stampa. Inizialmente si può fare qualche esperimento per scoprire in che punto conviene inserire le proprie modifiche. Tuttavia, se il contenuto del file PostScript è compresso, è praticamente impossibile insinuarsi nel suo codice. Nel caso fosse necessario espandere il codice PostScript, si può tentare di agire con il comando seguente:

```
$ gs -q -dNOPAUSE -dBATCH -sDEVICE=pswrite ↩
↪    -sOutputFile=output_file_ps ↩
↪    -c save pop -f input_file_ps [Invio]
```

Dove ***input_file_ps*** è il nome del file originale da espandere, mentre ***output_file_ps*** è il nome di quello che si vuole ottenere con la trasformazione.

Osservando un file PostScript, generato da un programma comune per la stampa, eventualmente dopo il procedimento di espansione appena mostrato, si dovrebbe individuare facilmente la fine di una pagina cercando l'istruzione '**showpage**':

```
PDFVars/TermAll get exec end end
userdict /pgsave get restore
showpage
%%PageTrailer
%%EndPage
```

Normalmente, subito prima di questa istruzione appare anche l'istruzione '**restore**' che serve a ripristinare le condizioni normali. Se effettivamente si recuperano tutte le impostazioni predefinite del linguaggio PostScript si può inserire subito prima dell'istruzione '**showpage**' qualcosa che si vuole sovrascrivere sulla pagina; per esempio un codice simile a quello seguente:

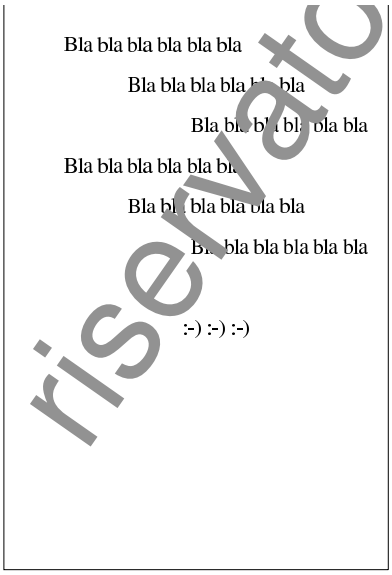
```
PDFVars/TermAll get exec end end
userdict /pgsave get restore
save
/cm { 28.346456 mul } def
   0 cm   0   cm moveto
  21 cm  29.7 cm lineto
   0 cm  29.7 cm moveto
  21 cm   0   cm lineto
stroke
restore
showpage
%%PageTrailer
%%EndPage
```

In questo caso si immagina di avere a disposizione una pagina A4, sulla quale si sovrascrive una «X» con due linee tra gli angoli opposti. Nel prossimo esempio, si intende sovrapporre la scritta «riservato» che appare in modo obliquo sulla pagina:

```
PDFVars/TermAll get exec end end
userdict /pgsave get restore
save
/cm { 28.346456 mul } def
    55  rotate
    7 cm  0 cm moveto
    /Helvetica findfont 7 cm scalefont setfont
    (riservato)
    0.5 setgray
    show
stroke
restore
showpage
%%PageTrailer
%%EndPage
```

La figura successiva mostra una pagina con un contenuto senza significato, sulla quale appare la scritta sovrapposta:

Figura 48.108. Esempio di una pagina con la sovrapposizione della scritta «riservato» in primo piano.

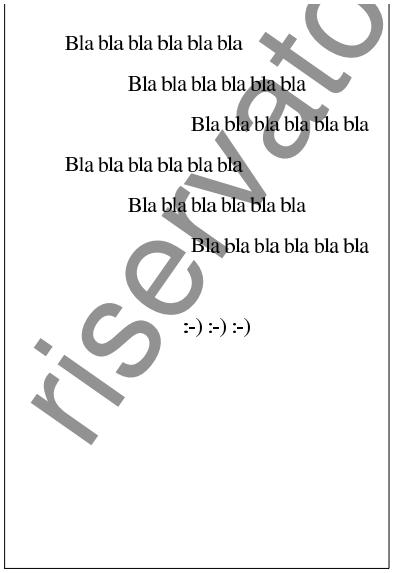


Se invece quello che si vuole è semplicemente di intervenire sullo sfondo della pagina, al posto di inserire il codice immediatamente prima dell'istruzione '**showpage**' occorre farlo all'inizio della pagina, precisamente subito dopo l'istruzione speciale '**%%Page:**', per esempio così:

```
%%Page: 1 10
save
/cm { 28.346456 mul } def
    55 rotate
    7 cm   0   cm moveto
    /Helvetica findfont 7 cm scalefont setfont
    (riservato)
    0.5 setgray
    show
stroke
restore
585 600 bop 1319 39 a Ii(Appunti)37 b(di)d(inf)-5
b(or)t(m)m(atica)35 b(liber)m(a)105 b Ic(2003.01.01)3042 243
y Ii(V)-13 b(olume)71
```

Nella figura successiva si mostra nuovamente l'esempio di una scritta obliqua che questa volta viene posta sullo sfondo, senza disturbare così la lettura del documento.

Figura 48.110. Esempio di una pagina con la scritta «riservato» sullo sfondo.



Eventualmente, per evitare sorprese, prima di modificare un file PostScript lo si può rielaborare attraverso '**ps2ps**'.

48.17.2 Due programmi Perl che fanno tutto da soli

Per risolvere il problema della modifica di file PostScript si può realizzare un piccolo programma Perl che legge un file contenente le aggiunte, filtrando un file PostScript:

```
ps-overwrite.pl file < file_ps_originale > file_ps_sovrascritto
```

```
ps-background.pl file < file_ps_originale > file_ps_sovrascritto
```

Per esempio, se il file '`aggiunte`' contiene il codice PostScript da aggiungere, i due comandi seguenti servirebbero per generare il file '`prova-modificato.ps`' dal file '`prova.ps`'; il primo per aggiungere il codice in primo piano, il secondo per aggiungerlo sullo sfondo:

```
$ cat prova.ps | ps-overwrite.pl aggiunte ↩
↪> prova-modificato.ps [Invio]

$ cat prova.ps | ps-background.pl aggiunte ↩
↪> prova-modificato.ps [Invio]
```

Ecco il primo programma:

```
#!/usr/bin/perl
local ($external_postscript_file) = $ARGV[0];
local ($external_postscript_code) = "";
local ($line)                     = "";

# Carica dal file esterno il codice da aggiungere.
open (PSCODE, "< $external_postscript_file");
while ($line = <PSCODE>)
  {
    $external_postscript_code = $external_postscript_code . $line;
  }
close (PSCODE);

# Legge lo standard input e cerca l'istruzione "showpage" per
# inserire il codice aggiunto.
while ($line = <STDIN>)
  {
    if ($line =~ m/^\%/)
      {
        # Si tratta di un commento o di un'istruzione speciale.
        print STDOUT ($line);
      }
    elsif ($line =~ m/^(.*)\bshowpage\b(.*)$/)
      {
        # Si aggiunge il codice.
        print STDOUT ($1);
        print STDOUT ("\n");
        print STDOUT ("save");
        print STDOUT ("\n");
        print STDOUT ("$external_postscript_code");
        print STDOUT ("\n");
        print STDOUT ("restore");
        print STDOUT ("\n");
        print STDOUT ("showpage");
        print STDOUT ("\n");
        print STDOUT ($2);
        print STDOUT ("\n");
      }
    else
      {
        print STDOUT ($line);
      }
  }
```

Ecco il secondo programma:

```
#!/usr/bin/perl
local ($external_postscript_file) = $ARGV[0];
local ($external_postscript_code) = "";
local ($line)                     = "";

# Carica dal file esterno il codice da aggiungere.
open (PSCODE, "< $external_postscript_file");
while ($line = <PSCODE>)
  {
    $external_postscript_code = $external_postscript_code . $line;
  }
close (PSCODE);

# Legge lo standard input e cerca l'istruzione speciale "%%Page:" per
# inserire il codice aggiunto.
while ($line = <STDIN>)
  {
    if ($line =~ m/^\%\%Page\:\s/)
      {
        # Si aggiunge il codice a partire dalla riga successiva.
        print STDOUT ($line);
        print STDOUT ("save");
        print STDOUT ("\n");
        print STDOUT ("$external_postscript_code");
        print STDOUT ("\n");
        print STDOUT ("restore");
        print STDOUT ("\n");
      }
    else
      {
        print STDOUT ($line);
      }
  }
```

Si può osservare che il codice aggiunto viene contornato automaticamente dalle istruzioni '**save**' e '**restore**' per maggior sicurezza.

48.17.3 Utilizzo pratico di questi programmi

Per fare in modo che tutte le stampe vengano alterate con qualche tipo di informazione, si deve intervenire nella gestione delle code di stampa. Se si dispone di un sistema di stampa BSD o compatibile, si può agire nel file '`/etc/printcap`', aggiungendo delle code differenti per ogni tipo di modifica che si vuole apportare; a ognuna di queste si deve abbinare un filtro differente.

```
lp|HP Laserjet:\
        :lp=/dev/lp0:\
        :sd=/var/spool/lpd/lp:\
        :af=/var/log/lp-acct:\
        :lf=/var/log/lp-errs:\
        :if=/etc/magicfilter/ljet4-filter:\
        :pl#66:\
        :pw#80:\
        :pc#150:\
        :mx#0:\
        :sh:
```

L'esempio che si vede sopra fa riferimento a una direttiva della coda di stampa predefinita, a cui si associa il filtro '`/etc/magicfilter/ljet4-filter`'. Si tratta precisamente di un programma scritto per Magicfilter (sezione 27.5), il quale contiene in particolare le direttive seguenti:

```
# PostScript
0       %!      filter  /usr/bin/gs  -q -dSAFER
-dNOPAUSE -r600 -sDEVICE=ljet4 -sOutputFile=- -
0       \004%!  filter  /usr/bin/gs  -q -dSAFER
-dNOPAUSE -r600 -sDEVICE=ljet4 -sOutputFile=- -
```

A questo punto si potrebbero scrivere una serie di file contenenti codice PostScript da aggiungere alle stampe, a seconda delle esigenze. Si suppone di collocare questi file nella directory '`/etc/ps-modifica/`'. Si osservi il listato successivo che si riferisce ad altre due code di stampa speciali:

```
logo|HP Laserjet con sovrascrittura del logo aziendale:\
        :lp=/dev/lp0:\
        :sd=/var/spool/lpd/lp:\
        :af=/var/log/lp-acct:\
        :lf=/var/log/lp-errs:\
        :if=/etc/magicfilter/ljet4-logo-filter:\
        :pl#66:\
        :pw#80:\
        :pc#150:\
        :mx#0:\
        :sh:

fatt|HP Laserjet con fincatura delle fatture:\
        :lp=/dev/lp0:\
        :sd=/var/spool/lpd/lp:\
        :af=/var/log/lp-acct:\
        :lf=/var/log/lp-errs:\
        :if=/etc/magicfilter/ljet4-fattura-filter:\
        :pl#66:\
        :pw#80:\
        :pc#150:\
        :mx#0:\
        :sh:
```

A questo punto si prende il file '`/etc/magicfilter/ljet4-filter`' e se ne fanno due copie: rispettivamente '`/etc/magicfilter/ljet4-logo-filter`' e '`/etc/magicfilter/ljet4-fattura-filter`'. All'interno di questi file si devono modificare le direttive riferite al filtro PostScript; in questo caso si mostra l'uso dello script '**`ps-overwrite`**', per la sovrapposizione in primo piano, ma l'uso dell'altro script è identico:

```
# PostScript
0       %!      filter  /usr/bin/ps-overwrite
/etc/ps-modifica/logo | /usr/bin/gs  -q -dSAFER -dNOPAUSE
-r600 -sDEVICE=ljet4 -sOutputFile=- -
0       \004%!  filter  /usr/bin/ps-overwrite
/etc/ps-modifica/logo | /usr/bin/gs  -q -dSAFER -dNOPAUSE
-r600 -sDEVICE=ljet4 -sOutputFile=- -
```

```
# PostScript
0       %!      filter  /usr/bin/ps-overwrite
/etc/ps-modifica/fattura
| /usr/bin/gs  -q -dSAFER -dNOPAUSE
-r600 -sDEVICE=ljet4 -sOutputFile=- -
0       \004%!  filter  /usr/bin/ps-overwrite
/etc/ps-modifica/fattura
| /usr/bin/gs  -q -dSAFER -dNOPAUSE
-r600 -sDEVICE=ljet4 -sOutputFile=- -
```

Se si vuole aggiungere anche un passaggio con '**`ps2ps`**':

```
# PostScript
0       %!      filter  /usr/bin/ps2ps - -
| /usr/bin/ps-overwrite /etc/ps-modifica/logo
| /usr/bin/gs  -q -dSAFER -dNOPAUSE
-r600 -sDEVICE=ljet4 -sOutputFile=- -
0       \004%!  filter  /usr/bin/ps2ps - -
| /usr/bin/ps-overwrite /etc/ps-modifica/logo
| /usr/bin/gs  -q -dSAFER -dNOPAUSE
-r600 -sDEVICE=ljet4 -sOutputFile=- -
```

```
# PostScript
0       %!      filter  /usr/bin/ps2ps - -
| /usr/bin/ps-overwrite /etc/ps-modifica/fattura
| /usr/bin/gs  -q -dSAFER -dNOPAUSE
-r600 -sDEVICE=ljet4 -sOutputFile=- -
0       \004%!  filter  /usr/bin/ps2ps - -
| /usr/bin/ps-overwrite /etc/ps-modifica/fattura
| /usr/bin/gs  -q -dSAFER -dNOPAUSE
-r600 -sDEVICE=ljet4 -sOutputFile=- -
```

Naturalmente rimangono da preparare i file '`/etc/ps-modifica/logo`' e '`/etc/ps-modifica/fattura`' con il codice PostScript che si preferisce.

Alla fine, è sufficiente selezionare una coda o l'altra per aggiungere il logo o la fincatura di una fattura:

```
$ lpr -Plogo ... [ Invio ]

$ lpr -Pfatt ... [ Invio ]
```

48.18 Riferimenti

- *Cappella Archive*, http://www.cappella.demon.co.uk/index.html
- David Byram-Wigfield, *Practical PostScript*, http://www.cappella.demon.co.uk/bookpdfs/pracpost.pdf
- David Byram-Wigfield, *Making an electronic book*, http://www.cappella.demon.co.uk/tinypdfs/06eb06ook.pdf
- Paul Bourke, *PostScript Tutorial*, http://local.wasp.uwa.edu.au/~pbourke/dataformats/postscript/
- *First Guide to PostScript*, http://www.tailrecursive.org/postscript/postscript.html
- *Portable Document Format Reference Manual*, 1999, *http://www.hamburg.baw.de/docs/pdfspec.pdf* (non più disponibile)

Capitolo 49

TeX

TeX è un linguaggio di programmazione per la composizione tipografica. La «compilazione» di un sorgente TeX produce un file in formato finale (DVI o PDF) per la stampa.

L'importanza di questo linguaggio richiede anche la conoscenza della pronuncia corretta del suo nome: «t-e-k». Infatti, il creatore di questo linguaggio, D.E. Knuth, voleva fare riferimento alle lettere greche maiuscole «TEX», che così vanno pronunciate.

Dal momento che le istruzioni TeX utilizzano spesso le parentesi graffe, nei modelli sintattici queste vanno intese letteralmente, come parte dell'istruzione rappresentata.

49.1 La distribuzione Unix di TeX

TeX è un linguaggio di programmazione per l'editoria elettronica. Come nei linguaggi di programmazione comuni è possibile realizzare procedure o funzioni, con TeX è possibile costruire delle macro. Nel tempo sono stati realizzati diversi pacchetti standard di macro per TeX; per esempio LaTeX e AmS-TeX.

Semplificando le cose, una distribuzione TeX è un insieme composto da un compilatore TeX (ma forse è più appropriato il termine «compositore»), una serie di file contenenti le informazioni necessarie a produrre i caratteri da stampa e alcuni pacchetti di macro (di solito si tratta almeno di LaTeX).

49.1.1 Collocazione

È importante chiarire che non esiste un modo standard di installare una distribuzione TeX e ogni distribuzione GNU tende a collocarla dove ritiene più opportuno. Il blocco principale di una distribuzione TeX dovrebbe trovarsi in una gerarchia che può inserirsi al di sotto di '`/usr/lib/`' o '`/usr/share/`'. A titolo di esempio, viene mostrato un elenco di alcune di queste possibilità.

'`/usr/lib/teTeX/texmf/`'

'`/usr/lib/texmf/texmf/`'

'`/usr/share/teTeX/texmf/`'

'`/usr/share/texmf/`'

Negli esempi che vengono mostrati, quando si fa riferimento a questa directory, si indicano solo percorsi relativi a iniziare da '`texmf/`'. La sigla «texmf» sta per *TeX and more*, oppure per *TeX and friends*.

49.1.2 Configurazione

Di fronte alla complicazione di una distribuzione TeX, potrebbe sembrare assurda l'idea di metterci le mani, pensando addirittura di modificare le impostazioni generali di TeX. Tuttavia, quando si maneggiano documenti eccezionalmente voluminosi, potrebbe essere necessario modificare anche ciò che non è stato pensato per esserlo.

Alla fine della composizione di un documento TeX, si può leggere nel file delle registrazioni generato, un rapporto delle risorse utilizzate durante l'elaborazione. Si osservi l'esempio.

```
Here is how much of TeX's memory you used:
 2418 strings out of 25906
 45132 string characters out of 446921
 109255 words of memory out of 263001
 5196 multiletter control sequences out of 10000+0
 106774 words of font info for 69 fonts, ↩
↪out of 200000 for 1000
 15 hyphenation exceptions out of 1000
 33i,12n,21p,2494b,1259s stack positions ↩
↪out of 300i,100n,500p,30000b,4000s

Output written on texput.dvi (1844 pages, 7563800 bytes).
```

Questo è proprio il caso di un documento enorme (1844 pagine), ma prima di tale informazione appaiono dei valori, dove alternativamente si vede quanto di una data risorsa è stato usato e quanto di questa sarebbe stato invece disponibile. Se per qualche ragione si esaurisce una di queste risorse, l'elaborazione si interrompe con una segnalazione di errore che indica quale limite è stato superato.

Se succede, si può provare a mettere mano al file di configurazione della distribuzione di TeX che dovrebbe essere '`texmf/web2c/texmf.cnf`'. La prima volta, non è tanto facile capire il senso delle direttive che questo contiene, ma con un po' di tentativi si dovrebbe riuscire a risolvere il problema.

Prima di tutto si può osservare che, seguendo lo stile generale di TeX, i commenti sono introdotti dal simbolo di percentuale ('`%`'). Nella prima parte del file sono annotati i percorsi dei vari componenti della distribuzione.

```
% Part 1: Search paths and directories.

% The main tree, which must be mentioned in $TEXMF, below:
TEXMFMAIN = /usr/share/texmf

% A place for local additions to a "standard" texmf tree.
% For example:
%   TEXMFLOCAL = /usr/share/texmf.local

% A place where texconfig stores modifications (instead of
% the TEXMFMAIN tree). texconfig relies on the name, so
% don't change it.
%   TEXMF_CNF = $TEXMF.cnf

% User texmf trees can be catered for like this...
%   HOMETEXMF = $HOME/texmf
```

La lettura di questa parte può rivelare delle informazioni importanti riguardo la propria distribuzione TeX. Più avanti inizia una parte più delicata: quella che definisce le dimensioni degli array utilizzati da TeX, dimensioni che di conseguenza rappresentano i limiti a cui si accennava all'inizio di questa sezione.

```
% Part 3: Array and other sizes for TeX (and Metafont and
% MetaPost).
…
% Max number of characters in all strings, including all
% error messages, help texts, font names, control sequences.
%  These values apply to TeX and MP.
pool_size.context = 500000
pool_size.cont-en = 500000
pool_size.cont-nl = 500000
pool_size.cont-de = 500000
%pool_size = 125000
pool_size = 500000
```

In questa parte, il valore più importante è quello di '**`pool_size`**', perché può creare problemi soprattutto a pdfTeX. Nell'esempio si vede che è stato quadruplicato.

Alcune modifiche non possono essere prese in considerazione senza un'elaborazione successiva del file. In generale, al termine delle modifiche è bene dare il comando seguente:

```
# texconfig init [Invio]
```


A parte il caso particolare dell'utilizzo appena mostrato, '**`texconfig`**' è un programma interattivo predisposto per configurare gli elementi essenziali della distribuzione TeX. Si avvia semplicemente, senza bisogno di argomenti. La figura 49.4 mostra il menù principale di '**`texconfig`**'.

```
# texconfig [Invio]
```

Figura 49.4. Il menù principale di '**`texconfig`**'.

```
.------------------ teTeX setup utility -------------------.
|                                                           |
|  Hint: all output of external commands (e.g. tex) is      |
|  logged into a file. You can look at this file using      |
|  LOG. If cursor keys make trouble, you may have more      |
|  luck with +/- and TAB.                                   |
| .-------------------------------------------------------. |
| |      EXIT    exit                                     | |
| |      PREF    personal preferences                     | |
| |      CONF    show configuration                       | |
| |      REHASH  rebuild ls-R database                    | |
| |      HYPHEN  hyphenation table (tex/latex)            | |
| |      MODE    default mode (xdvi/dvips/mf)             | |
| |      XDVI    xdvi configuration                       | |
| |      DVIPS   dvips configuration                      | |
| |      FONT    directories for font creation            | |
| |      DOC     rebuild html documentation               | |
| |      FAQ     frequently asked questions + answers     | |
| |      LOG     view logfile                             | |
|-------------------------------------------------------------|
|                  <  OK  >      <Cancel>                   |
'-------------------------------------------------------------'
```

La funzione indicata con la sigla *PREF* serve solo a modificare il comportamento di '**`texconfig`**', permettendo in particolare di selezionare un programma per la modifica del testo alternativo a quello predefinito.

La funzione *CONF* permette di mostrare la configurazione, consistente nella definizione delle directory contenenti i vari componenti della distribuzione TeX.

La funzione *REHASH* permette di ricostruire il file '`texmf/ls-R`', utilizzato per agevolare la ricerca dei componenti installati ad alcuni programmi della distribuzione. In generale, si ricostruisce questo file quando si aggiunte o si toglie qualche file (per esempio i file dei tipi di carattere).

La funzione *HYPHEN* è molto importante, perché permette di stabilire le lingue per cui attivare la suddivisione in sillabe del testo. Selezionando questa funzione si ottiene l'avvio del programma per la modifica di file di testo (presumibilmente VI) con il file '`texmf/tex/generic/config/language.dat`'. Questo file può essere modificato, quindi, dopo averlo salvato, vengono avviate automaticamente le procedure necessarie ad attivare in pratica le scelte fatte.

Di solito, si tratta di commentare le righe che fanno riferimento a linguaggi che non si vogliono gestire e di togliere il commento dalla direttiva di attivazione della sillabazione per la lingua italiana.

```
% File    : language.dat
% Purpose : specify which hypenation patterns to load
%           while running iniTeX

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% CAUTION: the first language will be the default if no
%          style-file (e.g. german.sty) is used.
% Since version 3.0 of TeX, hyphenation patterns for
% multiple languages are possible. Unless you know what you
% are doing, please let the american english patterns be the
% first ones. The babel system allows you to easily change
% the active language for your texts. For more information,
% have a look to the documentation in
% texmf/doc/generic/babel.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% The hyphenation pattern files are in the directory:
%       texmf/tex/generic/hyphen
```

```
% The US-english patterns should be loaded *always* and as
% *first* ones.
american ushyph1.tex %

% Let us define USenglish as an alias for american:
=USenglish %

% UK english, TWO LINES!
%british  ukhyph.tex %
%=UKenglish %

% english should always be defined. Either an alias for
% american or british.
=english %

% French, TWO lines!
french          frhyph.tex frhyphex.tex %
=patois %

german          ghyph31.tex %

% The following languages are disabled by default.
% Uncomment, what you need.
%bahasa         inhyph.tex %
%catalan        cahyph.tex %
%croatian       hrhyph.tex %
%czech          czhyph2e.tex %
%danish         dkhyphen.tex %
%dutch          nehyph2.tex %
%finnish        fihyph.tex %
%galician       gahyph.tex %
italian         ithyph.tex %
%magyar         huhyph.tex %
%norsk          nohyph.tex %
%polish         plhyph.tex %
%portuges       pthyph.tex %
%romanian       rohyphen.tex %
%russian        ruhyph.tex %
%serbocroatian  shhyphl.tex %
%slovene        sihyph22.tex %
%spanish        sphyph.tex %
%swedish        sehyph.tex %
%turkish        trhyph.tex %

% A "language" without hyphenation:
nohyphenation zerohyph.tex %
```

Al termine dell'elaborazione si può verificare nel file '`texmf/web2c/latex.log`' la presenza delle righe che dimostrano l'abilitazione della sillabazione per le lingue selezionate nel file di configurazione. In questo caso particolare, la lingua italiana corrisponde al linguaggio numero tre.

```
…
\l@american=\language0
…
\l@USenglish =\language0
\l@english =\language0
\l@french=\language1
…
\l@patois =\language1
\l@german=\language2
…
\l@italian=\language3
…
\l@nohyphenation=\language4
```

La funzione *MODE* permette di predisporre alcuni programmi per la risoluzione della propria stampante. Ciò si ottiene semplicemente selezionando il nome di una stampante che dovrebbe corrispondere, o essere molto simile alla propria.

La funzione *XDVI* permette di configurare Xdvi (sezione 26.3), il programma di visualizzazione dei file DVI, in modo da stabilire il formato del foglio che si utilizza. Basta scorrere un elenco e confermare.

La funzione *DVIPS* permette di configurare Dvips (sezione 26.3), il programma in grado di convertire file DVI in PostScript. Come per

Xdvi, la cosa più importante da stabilire è il formato della carta, ma può anche essere indicata la stampante, o il comando di stampa da utilizzare quando Dvips viene usato per inviare direttamente un file alla stampa.

L'ultima funzione importante è *FONT* che permette di regolare i permessi di accesso alle directory che contengono i tipi di carattere e anche di configurare altre caratteristiche di questi file.

49.1.3 Funzionamento fondamentale di TeX

TeX utilizza un sorgente che si distingue perché di solito il suo nome finisce con l'estensione '`.tex`'; durante il processo di composizione genera un rapporto sull'elaborazione in un file con l'estensione '`.log`' e produce un file finale in formato DVI, con estensione '`.dvi`'. Successivamente, i file DVI vengono convertiti normalmente in PostScript attraverso il programma Dvips.

Eventualmente, è disponibile anche pdfTeX, con cui, invece di una composizione in formato DVI, si ottiene un file PDF senza passaggi intermedi.

Se si suppone che il file '`primo.tex`' contenga il testo seguente

```
Ciao a tutti.

Questo \'e il mio primo documento scritto con il
linguaggio \TeX.
\bye
```

per ottenere la composizione in formato DVI è sufficiente il comando

```
$ tex primo.tex [Invio]

 This is TeX, Version 3.14159 (Web2C 7.3.1)
 (primo.tex [1] )
 Output written on primo.dvi (1 page, 328 bytes).
 Transcript written on primo.log.
```

mentre per ottenere la composizione in formato PDF è sufficiente il comando

```
$ pdftex primo.tex [Invio]

 This is pdfTeX, Version 3.14159-13d (Web2C 7.3.1)
 (primo.tex[/usr/share/texmf/pdftex/config/pdftex.cfg]
 Babel <v3.6x> and hyphenation patterns for american,
    italian, nohyphenation, loaded.
 [1[/usr/share/texmf/dvips/config/pdftex.map]] )<cmr10.pfb>
 Output written on primo.pdf (1 page, 9807 bytes).
 Transcript written on primo.log.
```

Nel primo caso si ottiene il file '`primo.dvi`', mentre nel secondo si ha il file '`primo.pdf`'. Eventualmente, per convertire il file DVI in PostScript, è sufficiente usare Dvips nel modo seguente:

```
$ dvips -o primo.ps primo.dvi [Invio]

 This is dvips(k) 5.86 Copyright 1999 Radical Eye Software
   (www.radicaleye.com)
 ' TeX output 2001.08.30:0835' -> primo.ps
 <texc.pro>. [1]
```

Ecco quello che si ottiene:

Ciao a tutti.
Questo è il mio primo documento scritto con il linguaggio TeX.

Si può anche usare una versione estesa di TeX, e-TeX e pdfeTeX, corrispondenti agli eseguibili '**`etex`**' e '**`pdfetex`**', che in questo caso si comportano nello stesso modo:

```
$ etex primo.tex [Invio]

 This is e-TeX, Version 3.14159-2.1 (Web2C 7.3.1)
 entering extended mode
 (primo.tex [1] )
 Output written on primo.dvi (1 page, 328 bytes).
 Transcript written on primo.log.
```

oppure

```
$ pdfetex primo.tex [Invio]
```

```
This is pdfeTeX, Version 3.14159-13d-2.1 (Web2C 7.3.1)
entering extended mode
(primo.tex[/usr/share/texmf/pdftex/config/pdftex.cfg] [1[
/usr/share/texmf/dvips/config/pdftex.map]] )<cmr10.pfb>
Output written on primo.pdf (1 page, 9807 bytes).
Transcript written on primo.log.
```

Gli eseguibili '**tex**', '**pdftex**' '**etex**' e '**pdfetex**' sono indipendenti, mentre attorno a loro si presentano altrettante serie di collegamenti simbolici:

```
virtex     -> tex
initex     -> tex
latex      -> tex
amstex     -> tex
evirtex    -> etex
einitex    -> etex
elatex     -> etex
pdfevirtex -> pdfetex
pdfeinitex -> pdfetex
pdfelatex  -> pdfetex
pdfvirtex  -> pdftex
pdfinitex  -> pdftex
pdflatex   -> pdftex
```

A seconda del nome usato per avviare uno stesso eseguibile, si può ottenere un comportamento differente. Nel caso di '**virtex**' che è un collegamento a '**tex**', si fa riferimento implicitamente al ***formato*** '**plain**', corrispondente alle dichiarazioni contenute nei file della directory '`texmf/tex/plain/`', così come '**pdfvirtex**' fa riferimento alla directory '`texmf/pdftex/plain/`', ecc. Purtroppo, le cose non sono così semplici in generale, perché le convenzioni non sono perfettamente omogenee; tuttavia, vale la pena di tenere presente che i nomi del tipo '**[pdf][e]virtex**' sono equivalenti ai nomi del tipo '**[pdf][e]tex**'.

I nomi del tipo '**[pdf][e]initex**' fanno riferimento al linguaggio TeX senza la dichiarazione di alcuna macro e sono equivalenti all'uso degli eseguibili del tipo '**[pdf][e]tex**' con l'aggiunta dell'opzione '**--ini**'. In pratica, per usare '**[pdf][e]initex**', oppure '**[pdf][e]tex --ini**', occorre modificare l'esempio già visto nel modo seguente:

```
\input plain
Ciao a tutti.

Questo \'e il mio primo documento scritto con il
linguaggio \TeX.
\bye
```

Questa spiegazione viene data solo per chiarire un po' il funzionamento di TeX e il significato di tutti i collegamenti simbolici che gli stanno intorno. L'uso dell'istruzione '**\input plain**' nel sorgente non funziona sempre come ci si aspetterebbe leggendo queste indicazioni; in pratica, una volta capito il senso della cosa, non va usata affatto.

In modo analogo a quanto visto fino a questo punto, quando si fa riferimento a un collegamento del tipo '**[pdf][e]latex**', è come usare un eseguibile del tipo '**[pdf][e]initex**', oppure '**[pdf][e]tex --ini**', iniziando il sorgente TeX con la riga seguente:

```
\input latex.ini
```

Resta il fatto che il comando '**\input latex.ini**' non rappresenta necessariamente uno standard e quello che conta è sapere solo che i collegamenti '**[pdf][e]latex**' richiamano in qualche modo il formato '**latex**'. In altri termini, i collegamenti '**[pdf][e]latex**' fanno riferimento implicitamente alle macro di LaTeX.

Per completare questa sezione, viene mostrato un esempio di un sorgente LaTeX, ovvero un sorgente TeX fatto per le macro LaTeX. Si fa riferimento al nome ipotetico '`secondo.tex`':


```
\documentclass{article}
\begin{document}
Ciao a tutti.

Questo \'e il mio primo documento scritto con il
linguaggio \LaTeX .
\end{document}
```

Per comporre correttamente questo file, occorre un comando del tipo:

```
$ [pdf][e]latex secondo.tex [Invio]
```

A seconda dei casi si ottiene il file '`secondo.dvi`', oppure '`secondo.pdf`'.

49.1.4 Interazione con TeX

La composizione di un documento scritto con il linguaggio TeX può avvenire anche con qualche forma di interazione. Se si avvia uno degli eseguibili '**[pdf][e]tex**' senza argomenti, si ottiene un invito a inserire il nome del file, attraverso l'indicazione di due asterischi:

```
$ tex [Invio]

 This is TeX, Version 3.14159 (Web2C 7.3.1)
 **
```

Si può inserire così il percorso del file, omettendo eventualmente l'estensione se questa corrisponde a '`.tex`':

```
**terzo.tex [Invio]

 (terzo.tex [1] )
 Output written on terzo.dvi (1 page, 488 bytes).
 Transcript written on terzo.log.
```

Supponendo di avere scritto un file, denominato '`quarto.tex`', in cui non appare l'istruzione '**\bye**' finale, come nel testo seguente, TeX si ferma in attesa di istruzioni, mostrando un invito ridotto a un solo asterisco:

```
Ciao a tutti.

Questo \'e il mio quarto documento scritto con il
linguaggio \TeX, dove non appare l'istruzione
$\backslash$bye alla fine del file.
```

```
$ tex quarto.tex [Invio]

 This is TeX, Version 3.14159 (Web2C 7.3.1)
 (quarto.tex)
 *
```

Naturalmente, se si è in grado di farlo, si può aggiungere anche altro testo:

```
*Saluti! [Invio]

*\bye [Invio]

 [1]
 Output written on quarto.dvi (1 page, 448 bytes).
 Transcript written on quarto.log.
```

Ciao a tutti.
Questo è il mio quarto documento scritto con il linguaggio TEX, dove non appare l'istruzione \bye alla fine del file.
Saluti!

Di solito si evita di interagire con TeX, tuttavia si può essere costretti dal presentarsi di un errore durante la compilazione del sorgente. Per la precisione, il livello di interazione di TeX può essere regolato attraverso delle istruzioni speciali, come descritto nella tabella 49.24. In condizioni normali, il funzionamento avviene in modalità '**errorstopmode**', corrispondente all'istruzione '**\errorstopmode**', in cui TeX si ferma in attesa di indicazioni per qualunque errore si presenti.

Tabella 49.24. Istruzioni per il controllo della modalità di funzionamento interattiva.

Istruzione TeX	Modalità di funzionamento
`\errorstopmode`	La composizione viene sospesa per qualunque errore.
`\scrollmode`	La composizione viene sospesa solo per gli errori più importanti.

Istruzione TeX	Modalità di funzionamento
`\nonstopmode`	La composizione viene sospesa solo in presenza di errori gravissimi.
`\batchmode`	La composizione viene sospesa solo in presenza di errori gravissimi e non si mostrano informazioni sullo schermo.

Per esempio, il file '`quinto.tex`' che contiene il testo seguente, usa erroneamente l'istruzione '**`\tex`**' al posto di '**`\TeX`**':

```
Ciao a tutti.

Questo \‘e il mio quinto documento scritto con il
linguaggio \tex, in cui si provoca volutamente un errore.
\bye
```

```
$ tex quinto.tex [Invio]

This is TeX, Version 3.14159 (Web2C 7.3.1)
(quinto.tex
! Undefined control sequence.
l.3 ...to documento scritto con il linguaggio \tex
                                                , in
?
```

Viene dichiarato sinteticamente il tipo di errore individuato, che in questo caso corrisponde a

```
! Undefined control sequence.
```

ovvero una sequenza di controllo indefinita. Nella riga successiva si indica il numero della riga in cui appare l'errore ('**`l.3`**' sta per *line* 3) con il pezzo di testo che arriva fino all'errore, mentre il pezzo successivo appare staccato, nella riga successiva.

```
l.3 ...to documento scritto con il linguaggio \tex
                                                , in
```

In pratica, secondo TeX l'errore è riferito esattamente alla stringa '**`\tex`**'. Sotto appare un invito composto da un punto interrogativo, con il quale TeX attende un'azione dell'utente. Si può rispondere con un altro punto interrogativo per avere l'elenco delle possibilità:

```
??[Invio]

Type <return> to proceed, S to scroll future error messages,
R to run without stopping, Q to run quietly,
I to insert something, E to edit your file,
1 or ... or 9 to ignore the next 1 to 9 tokens of input,
H for help, X to quit.
?
```

La tabella 49.30 descrive brevemente il significato e l'uso dei comandi disponibili, mostrando anche la modalità corrispondente quando una scelta coincide con la richiesta di cambiamento di questa.

Tabella 49.30. Interazione in presenza di un errore.

Tastiera	Istruzione TeX corrispondente	Effetto
[*h*][*Invio*]		Mostra la spiegazione del motivo dell'interruzione.
[*i*][*Invio*]		Richiede di inserire una correzione.
[*e*][*Invio*]		Avvia un programma per la modifica del file sorgente.
[*x*][*Invio*]		Arresta la composizione.
[*s*][*Invio*]	`\scrollmode`	Non si ferma più in presenza di piccoli errori.
[*r*][*Invio*]	`\nonstopmode`	Non si ferma più per alcun errore.
[*q*][*Invio*]	`\batchmode`	Non si ferma più per alcun errore e non mostra alcuna informazione.
[*Invio*]		Cerca di rimediare all'errore e continua la composizione.
[***n***][*Invio*]		Cerca di rimediare automaticamente ai prossimi ***n*** errori.

Nel caso dell'errore mostrato, si vuole provare a capire meglio di cosa si tratta, attraverso il comando [*h*][*Invio*]:


```
?h [Invio]

  The control sequence at the end of the top line
  of your error message was never \def’ed. If you have
  misspelled it (e.g., ‘\hobx’), type ‘I’ and the correct
  spelling (e.g., ‘I\hbox’). Otherwise just continue,
  and I’ll forget about whatever was undefined.
```

Dal momento che si può correggere facilmente l'errore, si richiede di poter inserire del testo sostitutivo:

```
?i [Invio]

insert>\TeX [Invio]

  [1] )
  Output written on quinto.dvi (1 page, 376 bytes).
  Transcript written on quinto.log.
```

Con il comando [*e*][*Invio*] si può avviare un programma per la modifica del file sorgente. Di solito si tratta di VI, ma si può intervenire nella variabile di ambiente ***TEXEDIT*** indicando il nome di un altro programma, usando le metavariabili '**`%d`**' e '**`%s`**' per indicare rispettivamente la riga contenente l'errore e il nome del file. Per esempio, rimanendo nel caso di VI, è possibile usare la sintassi seguente per raggiungere la riga ***n***-esima del file indicato:

```
vi -c n file
```

In questo modo, si dovrebbe assegnare alla variabile di ambiente ***TEXEDIT*** la stringa '**`vi -c %d %s`**':

```
$ TEXEDIT="vi -c %d %s" [Invio]

$ export TEXEDIT [Invio]
```

49.2 Elementi essenziali del linguaggio TeX

Un file sorgente TeX è un file di testo normale; per la precisione dovrebbe trattarsi di un file ASCII standard a 7 bit, dove l'interruzione delle righe avviene secondo le regole del proprio sistema operativo.

Nel linguaggio di TeX si distinguono righe bianche o vuote da righe contenenti istruzioni. Nell'ambito delle righe contenenti istruzioni, possono poi apparire dei commenti che si distinguono per essere preceduti dal segno di percentuale ('**`%`**'), terminati dalla fine della riga. Nell'esempio seguente si può osservare che tutte le righe che contengono del testo sono in pratica delle istruzioni, più o meno articolate. Nella prima riga appare anche un commento che poi scompare nella composizione finale.

```
Ciao a tutti. % Ecco, un commento tanto per gradire.

Questo \‘e solo un piccolo esempio per vedere come funziona
il linguaggio di composizione \TeX.
\bye
```

Gli spazi, verticali e orizzontali, hanno un significato, ma generalmente non si sommano. Normalmente si usa una riga vuota o bianca per separare il testo in paragrafi, ma la presenza di più righe bianche o vuote non cambia la distanza tra questi paragrafi nella composizione finale; nello stesso modo, uno spazio orizzontale serve generalmente a separare le parole di una frase, ma la presenza di più spazi orizzontali non cambia la distanza tra le parole.

Eventualmente si può dichiarare espressamente la separazione tra i paragrafi attraverso l'istruzione '**`\par`**', tenendo presente che anche in questo caso, l'uso di più istruzioni del genere non produce una separazione maggiore tra i paragrafi stessi

L'istruzione finale '**`\bye`**' conclude il «programma» TeX e tutto ciò che appare dopo viene ignorato.

49.2.1 Testo letterale, parole e simboli di controllo

Una riga di testo si traduce generalmente nel testo corrispondente nella composizione finale, tenendo conto però che alcuni simboli hanno un significato speciale e si distinguono contesti differenti. La tabella 49.34 elenca i caratteri che hanno significati particolari, con

l'indicazione del modo per ottenere il loro simbolo originale nella composizione.

Tabella 49.34. Elenco dei caratteri che hanno significati particolari.

Carattere speciale	Utilizzo normale	Trasformazione per l'uso letterale
\	Prefisso di una parola di controllo o di un simbolo di controllo.	`$\backslash$`
{	Apre un gruppo.	`$\{$`
}	Chiude un gruppo.	`$\}$`
%	Inizia un commento fino alla fine della riga.	`\%`
&	Tabulazione orizzontale.	`\&`
~	Spazio non interrompibile.	`\~{}`
$	Inizia e conclude un contesto matematico.	`\$`
^	Apice, in un contesto matematico.	`\^{}`
_	Pedice, in un contesto matematico.	`\_{}`
#	Definisce la collocazione di un parametro.	`\#`
<*SP*>	Uno o più spazi vengono ridotti a uno solo.	`\<SP>`
<	In condizioni normali, genera un punto esclamativo rovesciato.	`$<$`
>	In condizioni normali, genera un punto interrogativo rovesciato.	`$>$`

La barra obliqua inversa ('\') viene usata come prefisso per delle ***sequenze di controllo***, nell'ambito delle quali si può distinguere tra ***parole di controllo*** e ***simboli di controllo***. Una parola di controllo è formata esclusivamente da lettere alfabetiche (dalla «a» alla «z», maiuscole e minuscole, escluse le lettere accentate); per esempio, '**\TeX**' è una parola di controllo con cui si ottiene la rappresentazione del nome TeX secondo lo standard stabilito dal suo autore originale. Un simbolo di controllo è invece un solo carattere che non sia una lettera alfabetica; per esempio, '**\'**' è un simbolo di controllo con cui si ottiene l'aggiunta di un accento grave sopra il simbolo successivo.

In base al fatto che le parole di controllo si distinguono perché composte esclusivamente da lettere alfabetiche, si può porre il problema di delimitarle correttamente quando si trovano incorporate in parole che compongono il testo normale. Nell'esempio già mostrato, la parola di controllo '**\TeX**' si individua correttamente perché è seguita da un punto fermo, ovvero un simbolo che non è una lettera alfabetica, ma volendo scrivere un gioco di parole, come «il mio primo TeX-documento», sarebbe necessario usare uno strattagemma. Si osservi l'esempio seguente che si traduce in un errore nella composizione del documento:

```
Ciao a tutti.

Questo \'e il mio primo \TeXdocumento.
\bye
```

TeX cerca di interpretare la parola di controllo '**\TeXdocumento**' e non dovrebbe riuscirci. Per questa ragione, lo spazio che dovesse seguire una parola di controllo viene ignorato; così, diventa più facile inserire queste parole di controllo in parole del testo che si vuole comporre. Ecco l'esempio corretto:

```
Ciao a tutti.

Questo \'e il mio primo \TeX documento.
\bye
```

Tuttavia, rimane da chiarire in che modo inserire veramente uno spazio dopo una parola di controllo. Questo problema viene risolto con l'uso dei raggruppamenti.

49.2.2 Raggruppamenti

In diverse situazioni è utile raggruppare parte delle istruzioni (il testo) all'interno di parentesi graffe ('**{**…**}**'). L'effetto del raggruppamento non si nota nella composizione finale, ma permette di circoscrivere l'effetto di istruzioni particolari.

È ammissibile anche l'uso di raggruppamenti vuoti, '**{}**', che di solito vengono usati per separare parole o simboli di controllo dal testo che segue. Per esempio, scrivendo '**\TeX{}**' si riesce a evitare che lo spazio successivo venga inghiottito. Naturalmente si può usare anche il simbolo di controllo '****<***SP***>' (barra obliqua inversa e spazio) per indicare espressamente uno spazio in quel punto. Nel seguito vengono mostrati diversi esempi che si traducono nella stessa composizione finale.

- Raggruppamento vuoto alla fine della parola di controllo:

```
Il linguaggio di composizione \TeX{} sembra complesso
a prima vista...
\bye
```

- Raggruppamento vuoto per separare gli spazi:

```
Il linguaggio di composizione \TeX {} sembra complesso
a prima vista...
\bye
```

- Raggruppamento per individuare uno spazio:

```
Il linguaggio di composizione \TeX{ }sembra complesso
a prima vista...
\bye
```

- Raggruppamento per individuare uno spazio dopo la fine della parola di controllo:

```
Il linguaggio di composizione \TeX { }sembra complesso
a prima vista...
\bye
```

- Raggruppamento per isolare la parola di controllo:

```
Il linguaggio di composizione {\TeX} sembra complesso
a prima vista...
\bye
```

- Raggruppamento per isolare lo spazio e la parola successiva:

```
Il linguaggio di composizione \TeX { sembra} complesso
a prima vista...
\bye
```

Nella tabella 49.34 sono stati mostrati alcuni simboli di controllo che sono conclusi evidentemente da un raggruppamento vuoto: '**\~{}**', '**\^{}**' e '**_{}**'. In questo caso, il raggruppamento vuoto serve a impedire che la sequenza di controllo produca qualcosa di diverso da ciò che ci si aspetta in quel contesto particolare. Per esempio, '**\^o**' oppure anche '**\^ o**' genera la lettera accentata «ô».

Viene chiarito in seguito che le istruzioni '**\~{}**' e '**\^{}**' sono delle macroistruzioni che servono a generare un accento, utilizzano un parametro, costituito dal carattere o dal gruppo successivo. Nel momento in cui si vogliono rappresentare questi simboli senza abbinarli ad altro, il loro parametro deve essere un gruppo vuoto o lo spazio inteso come carattere ('****<***SP***>'). Pertanto, in alternativa a '**\~{}**' e '**\^{}**' si potrebbe usare la forma '**\~**<***SP***>' e '**\^**<***SP***>'.

Oltre all'uso delle parentesi, è possibile usare in alternativa la coppia di parole di controllo '**\bgroup**' e '**\egroup**', in sostituzione di '**{**' e '**}**' rispettivamente.

49.2.3 Inclusione di file esterni

È possibile articolare un sorgente TeX in più file separati che vengono inclusi con l'istruzione '**\input**':

```
\input nome_file
```

L'inclusione avviene inserendo in quel punto le righe del file indicato. Se il file esterno contiene l'istruzione '**\bye**', la composizione termina senza continuare nel file di partenza.

49.3 Variabili e tipi di dati

TeX ha una gestione molto particolare dei dati. Nelle sezioni seguenti vengono descritti solo i tipi di dati più comuni, ma questo dovrebbe bastare per far comprendere la logica di fondo.

Vale la regola per cui può essere usato solo ciò che è già stato dichiarato; inoltre, il campo di azione di queste variabili può essere controllato attraverso i raggruppamenti con le parentesi graffe.

In generale, per TeX una variabile ha l'aspetto di una parola di controllo a cui si assegna un valore secondo la sintassi seguente:

```
\nome=valore
```

L'espansione di una variabile avviene inserendo la parola di controllo corrispondente nel punto in cui il contesto lo richiede. Tuttavia, se lo scopo è quello di espandere la variabile in modo che appaia nel testo normale, occorre usare un accorgimento, con cui si trasforma il suo contenuto in una stringa. Di solito si usa per questo la parola di controllo '**\the**':

```
\the\nome
```

49.3.1 Stringhe

Per TeX la stringa è ciò che può essere reso tipograficamente; così, un'espressione stringa è ciò che alla fine si trasforma in una stringa nella composizione tipografica. Le variabili di tipo stringa si dichiarano nel modo seguente:

```
\newtoks\nome
```

Si può assegnare una stringa alla variabile nel modo seguente:

```
\nome={espressione_stringa}
```

L'espansione di una variabile stringa non può avvenire inserendo semplicemente la parola di controllo '***nome***' nel testo, perché occorre dichiarare espressamente questa intenzione con la parola di controllo '**\the**'.

```
\newtoks\data
\data={9 settembre 2001}
Treviso, \the\data \par
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. \par
Ciao.
```

Treviso, 9 settembre 2001
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.
Ciao.

Nell'esempio precedente, si crea una variabile stringa corrispondente alla parola di controllo '**\data**', a cui si assegna la stringa '**9 settembre 2001**', per poi ottenerne successivamente l'espansione nel testo.

49.3.2 Trasformazione delle stringhe

Le trasformazioni più comuni sulle stringhe sono il cambiamento in maiuscole o minuscole. Ciò si ottiene facilmente con le macro '**\uppercase**' e '**\lowercase**':

```
\uppercase{espressione_stringa_da_trasformare_in_maiuscole}
```

```
\lowercase{espressione_stringa_da_trasformare_in_minuscole}
```

49.3.3 Contatori

La variabile numerica più semplice di TeX è il contatore. Ne esistono due tipi: uno deve essere inizializzato subito, con un valore non negativo, l'altro no:


```
\countdef\nome_contatore=n
```

```
\newcount\nome_contatore
```

Il primo di questi due modelli riguarda il tipo di contatore che deve essere inizializzato in fase di dichiarazione. Il valore di inizializzazione è rappresentato da ***n***.

Per assegnare un valore a una variabile contatore, si usa la forma seguente:

```
\nome=valore
```

È importante sottolineare che l'inizializzazione di un contatore definito attraverso '**\countdef**' potrebbe in pratica non tradursi nell'assegnamento corrispondente alla variabile, pur essendo obbligatorio. Pertanto, conviene poi assegnare nuovamente il valore richiesto.

Si può assegnare un numero espresso usando cifre numeriche, con un segno anteriormente nel caso sia necessario, senza separatore decimale, come nell'esempio seguente, in cui si assegna al contatore '**\conteggio**' il valore -345.

```
\conteggio=-345
```

49.3.4 Lunghezze

Un tipo specifico di variabile numerica è adibita a contenere delle lunghezze. Per TeX, la lunghezza è un'informazione numerica particolare, riconducibile al concetto di variabile a virgola mobile di altri linguaggi. Si dichiara una lunghezza nel modo seguente:

```
\newdimen\nome_lunghezza
```

Una lunghezza è un valore che si può rappresentare in forma costante solo specificando l'unità di misura, composta da due lettere secondo lo schema che appare nella tabella 49.49. Pertanto, un valore che esprime una lunghezza deve avere la forma seguente:

```
[+|-]numero[true]unità_di_misura
```

In particolare, il valore che precede l'unità di misura può contenere una virgola decimale, espressa attraverso il punto ('.').[1] Per esempio, per esprimere una lunghezza di 10 cm, si deve scrivere '**10cm**'.[2] Nell'esempio seguente si assegna alla variabile '**\distanza**' una lunghezza positiva di 4,5 mm:

```
\distanza=4.5mm
```

La parola chiave '**true**' consente di indicare una lunghezza che non può essere ingrandita o ridotta attraverso l'istruzione '**\magnification**', come viene descritto nella sezione 49.11.1. Pertanto, se nell'esempio precedente si vuole indicare una lunghezza positiva corrispondente esattamente a 4,5 mm, in ogni situazione, si deve scrivere così:

```
\distanza=4.5truemm
```

oppure anche in modo più leggibile:

```
\distanza=4.5 true mm
```

La rappresentazione interna delle lunghezze è di un solo tipo; in pratica, TeX converte sempre i valori nell'unità di misura più piccola che è in grado di gestire. L'unità di misura più piccola è definita *scaled point* ed è stata creata appositamente per TeX.

Tabella 49.49. Unità di misura secondo TeX.

Sigla	Denominazione	Corrispondenza	Annotazioni
mm	millimetro		

Sigla	Denominazione	Corrispondenza	Annotazioni
`cm`	centimetro		
`in`	pollice	2,54 cm	
`bp`	*big point*	0,352 777 7 mm 1/72 pollici	Il punto tipografico usato dal linguaggio PostScript.
`pt`	punto	0,351 459 8 mm 1/72,27 pollici	Punto tipografico usato negli Stati Uniti.
`dd`	punto didôt	0,376 065 mm 1/67,54 pollici	Punto tipografico europeo.
`sp`	*scaled point*	1/65 535 punti '**`pt`**'	L'unità di misura più piccola gestibile da TeX.
`pc`	pica	12 punti	
`em`	M	variabile	Quadratone, pari alla larghezza della lettera «M» maiuscola.
`ex`	x	variabile	Altezza della lettera «x» minuscola.
`mu`	*math unit*	18 quadratoni	

Si osservi che TeX non semplifica la tradizione tipografica, consentendo di utilizzare ben tre tipi diversi di punto tipografico. Il punto a cui si è abituati comunemente con i programmi di composizione, è quello corrispondente alla sigla '**`bp`**', ma TeX utilizza in modo predefinito l'unità '**`pt`**', la quale comunque non si discosta di molto.

49.3.5 Lunghezze elastiche

In varie situazioni, TeX è in grado di gestire delle lunghezze elastiche. Le variabili che contengono informazioni del genere sono in grado di annotare tre indicazioni distinte: la distanza normale, una tolleranza in più e una tolleranza in meno. Questo tipo di informazione si esprime secondo la forma seguente:

```
lunghezza_richiesta [plus lunghezza_in_estensione] [minus lunghezza_in_contrazione]
```

In pratica, è come dire che si fa riferimento a una certa lunghezza, a cui si può aggiungere quanto appare dopo la parola chiave '**`plus`**' e si può togliere quanto appare dopo '**`minus`**'. Come si può intuire, quando non si indicano i valori che danno elasticità, si sottintende in corrispondenza un valore zero.

L'elasticità fissata attraverso le parole chiave '**`plus`**' e '**`minus`**' non è tassativa. Di solito, il solo fatto che si consenta un'estensione, anche di un solo punto, fa sì che il salto sia allungabile in modo indefinito, in caso di necessità.

Eventualmente, si dichiara una variabile del genere con la forma seguente:

```
\newskip\nome
```

L'assegnamento, come si può intendere, ha la forma seguente:

```
\nome=lunghezza_elastica
```

Ovvero:

```
\nome=lunghezza [plus lunghezza] [minus lunghezza]
```

Le indicazioni sull'elasticità in estensione e in contrazione sono formate normalmente da lunghezze, come per esempio '**`plus 1pt`**', ma si possono usare anche delle definizioni astratte, rappresentate da tre parole chiave, precedute da un numero intero:


```
nfil
nfill
nfilll
```

Generalmente, il numero ***n*** è sempre 1 e va inteso come moltiplicatore della parola successiva; in pratica, '**`2fil`**' rappresenta un'elasticità doppia di '**`1fil`**'.

La parola chiave '**`fil`**' rappresenta un'elasticità di grado minimo, '**`fill`**' un'elasticità di grado medio e '**`filll`**' un'elasticità molto grande.

49.3.6 Operazioni con i valori numerici

La realizzazione di espressioni numeriche con TeX diventa abbastanza complicata. Si utilizzano fondamentalmente tre tipi di istruzione per modificare il valore di una variabile:

```
\advance\nome [by] valore
```

```
\multiply\nome [by] valore
```

```
\divide\nome [by] valore
```

In pratica, nel primo caso si incrementa la variabile del valore indicato (se il valore in questione è negativo, la variabile viene ridotta di conseguenza); nel secondo caso si assegna alla variabile il prodotto tra quanto contenuto prima per il valore indicato; nel terzo caso si divide il contenuto della variabile per il valore, assegnando alla stessa il risultato della divisione.

Come si può osservare, la parola chiave '**`by`**' è facoltativa e si può usare per facilitare la lettura umana dell'istruzione.

Il valore usato deve essere del tipo adatto alla variabile con cui si esegue l'operazione. Viene mostrato un esempio complessivo che dovrebbe essere comprensibile a sufficienza:

```
\countdef\pagina=0
\pagina=1
\newcount\contatore
\contatore=-7
\newdimen\lunghezza
\lunghezza=100pt
\newskip\elastico
\elastico=10pt plus 2pt minus 1pt
pagina = \the\pagina \par
contatore = \the\contatore \par
lunghezza = \the\lunghezza \par
elastico = \the\elastico \par
\multiply\pagina by 2
\advance\contatore by 1
\advance\lunghezza by 10pt
\advance\elastico by 5pt plus 3pt minus 2pt
pagina = \the\pagina \par
contatore = \the\contatore \par
lunghezza = \the\lunghezza \par
elastico = \the\elastico \par
\divide\lunghezza by 2
\divide\elastico by 2
lunghezza = \the\lunghezza \par
elastico = \the\elastico \par
```

pagina = 1
contatore = -7
lunghezza = 100.0pt
elastico = 10.0pt plus 2.0pt minus 1.0pt
pagina = 2
contatore = -6
lunghezza = 110.0pt
elastico = 15.0pt plus 5.0pt minus 3.0pt
lunghezza = 55.0pt
elastico = 7.5pt plus 2.5pt minus 1.5pt

Va tenuta in considerazione una scorciatoia importante per rappresentare il prodotto tra una costante numerica e il valore di una lunghezza; questa scorciatoia si usa di solito per gli assegnamenti:

n**nome

Qui il numero ***n*** non deve esprimere una lunghezza, pertanto non può contenere l'indicazione dell'unità di misura. Per esempio, se '**\hsize**' contiene la lunghezza 15 cm, con l'istruzione seguente si assegna alla variabile '**\mezza**' la metà di questa lunghezza, ovvero 7,5 cm:

```
\mezza=0.5\hsize
```

Nell'esempio seguente si creano due variabili: una contiene una lunghezza e l'altra una lunghezza elastica. Dopo aver controllato il valore iniziale di queste, si riassegna loro la metà del loro valore di partenza, controllando successivamente il risultato ottenuto. Si può osservare che '**\elastico**' perde l'informazione sull'elasticità, diventando una lunghezza normale.

```
\newdimen\lunghezza
\lunghezza=100pt
\newskip\elastico
\elastico=10pt plus 2pt minus 1pt
lunghezza = \the\lunghezza \par
elastico = \the\elastico \par
\lunghezza=0.5\lunghezza
\elastico=0.5\elastico
lunghezza = \the\lunghezza \par
elastico = \the\elastico \par
```

lunghezza = 100.0pt
elastico = 10.0pt plus 2.0pt minus 1.0pt
lunghezza = 50.0pt
elastico = 5.0pt

49.3.7 Trasformazione in stringa di dati numerici

I dati numerici, quando devono essere convertiti in stringhe, possono essere rappresentati in forme differenti. La parola di controllo '**\the**' consente di ottenere una trasformazione «normale» di qualunque variabile in stringa:

\the*parola_di_controllo*

Per esempio, se '**\lunghezza**' è stata dichiarata come lunghezza contenente il valore 10 cm, l'espansione di '**\the\lunghezza**' può generare la stringa '**284.45274pt**'.

Un valore numerico può essere trasformato in numero intero, ammesso che ciò abbia senso, con la parola di controllo '**\number**':

\number*parola_di_controllo*

In questo caso, se si tenta di trasformare una lunghezza in numero, si ottiene la dimensione in punti *scaled point*; nel caso dell'esempio precedente, si otterrebbe la rappresentazione del valore 18646798, corrispondente ai 10 cm di prima espressi secondo l'unità di misura minima che TeX è in grado di gestire. Per quanto riguarda le lunghezze elastiche, non è possibile usare la trasformazione attraverso '**\number**'.

Quando il valore da trasformare è un intero positivo maggiore di zero, si può ottenere la rappresentazione in numero romano, con lettere minuscole, attraverso la parola di controllo '**\romannumeral**':

\romannumeral*parola_di_controllo*

L'esempio seguente riepiloga l'uso delle forme di trasformazione dei dati numerici in stringa che sono appena state descritte:

```
\newdimen\lunghezza
\newcount\contatore
\newcount\pagina
\lunghezza=1cm
\contatore=-7
\pagina=123
lunghezza = \the\lunghezza\ = \number\lunghezza scaled point \par
contatore = \the\contatore\ = \number\contatore \par
pagina = \the\pagina\ = \number\pagina = \romannumeral\pagina \par
```


lunghezza = 28.45274pt = 1864679scaled point
contatore = -7 = -7
pagina = 123 = 123= cxxiii

49.3.8 Controllo del campo di azione delle variabili

Le parentesi graffe, oppure le parole di controllo '**\bgroup**' e '**\egroup**', servono a delimitare e isolare una zona rispetto al testo che si trova al loro esterno. Una dichiarazione o un assegnamento fatto all'interno di una zona delimitata da parentesi graffe ha effetto in quell'ambito e in tutte le altre zone che possono essere annidate al suo interno, mentre all'esterno non esiste più. Si osservi l'esempio seguente:

```
\newdimen\lunghezza
\newcount\contatore
\lunghezza=1cm
\contatore=7
{
    \lunghezza=10cm
    \contatore=14
    {
        \lunghezza=100cm
        \contatore=21
        lunghezza = \the\lunghezza{}
        contatore = \the\contatore \par
     }
    lunghezza = \the\lunghezza{}
    contatore = \the\contatore \par
}
lunghezza = \the\lunghezza{} contatore = \the\contatore
\bye
```

Si ottiene il testo che segue:

lunghezza = 2845.27559pt contatore = 21
lunghezza = 284.52756pt contatore = 14
lunghezza = 28.45274pt contatore = 7

Perché un assegnamento abbia valore in modo globale, si usa la parola di controllo '**\global**'. Si osservi l'esempio seguente:

```
\newdimen\lunghezza
\newcount\contatore
\lunghezza=1cm
\contatore=7
{
    \lunghezza=10cm
    \contatore=14
    {
        \global\lunghezza=100cm
        \contatore=21
        lunghezza = \the\lunghezza{}
        contatore = \the\contatore \par
     }
    lunghezza = \the\lunghezza{}
    contatore = \the\contatore \par
}
lunghezza = \the\lunghezza{} contatore = \the\contatore
\bye
```

In questo caso, la lunghezza appare essere sempre di 100 cm (2845,27559 punti normali), anche quando si ritorna al di fuori dei raggruppamenti. In pratica, si ottiene il testo che segue:

lunghezza = 2845.27559pt contatore = 21
lunghezza = 2845.27559pt contatore = 14
lunghezza = 2845.27559pt contatore = 7

49.3.9 Ordine nell'espansione delle sequenze di controllo

Esiste un problema con TeX, legato all'ordine in cui vengono espanse le parole di controllo. L'esempio più comune è dato dalla difficoltà con cui si riesce a trasformare un numero romano ottenuto da '**\romannumeral**' in maiuscolo. Per esempio,

```
\newcount\pagina
\pagina=4
\uppercase{\romannumeral\pagina}
```

genera solo:

In pratica, '**\uppercase**' si trova a intervenire su una stringa che ancora non c'è, per cui non succede nulla. Per risolvere il problema si usa la parola di controllo '**\expandafter**' che anticipa l'esecuzione di ciò che segue:

```
\newcount\pagina
\pagina=4
\uppercase\expandafter{\romannumeral\pagina}
```

IV

Purtroppo il funzionamento di questa parola di controllo non è intuitivo e spesso si devono fare vari tentativi prima di riuscire a fare ciò che si intende.

49.4 Dichiarazione di macroistruzioni[3]

Si può dare un nome a un'espressione stringa attraverso la sintassi seguente:

```
\def\nome{espressione_stringa}
```

Si osservi che, al contrario di altre situazioni, qui TeX richiede che la parentesi graffa aperta segua immediatamente il nome (o il simbolo dell'ultimo parametro, come viene descritto nel seguito), senza alcuna spaziatura intermedia.

Questo tipo di dichiarazione serve in generale per realizzare delle macroistruzioni; tuttavia, utilizzandola solo così, si fa in modo di ottenere l'espansione di ciò che è contenuto fra le parentesi graffe nel momento in cui si inserisce nel testo l'istruzione '***nome***'. Si osservi l'esempio:

```
\def\eTeX{e-\TeX}

Bla bla bla bla \eTeX{} bla bla...
\bye
```

In questo caso si vuole dichiarare la parola di controllo '**\eTeX**', con cui diventa facile uniformare la scrittura di questo nome nel testo. Se ci fosse un ripensamento sulla forma da dare al nome, basterebbe modificare la sua dichiarazione iniziale.

Una macro del genere può essere modificata solo riassegnandole un altro valore, nello stesso modo usato per la sua dichiarazione iniziale.

È importante ricordare che in fase di composizione, le parole di controllo assorbono gli spazi alla loro destra, per cui è necessario usare una tecnica per evitarlo, come l'inserimento di un gruppo vuoto alla fine della stessa.

Nel momento in cui si utilizza una parola di controllo corrispondente a una macroistruzione dichiarata in questo modo, si ottiene l'espansione del suo contenuto. In altri termini, la parola di controllo diventa una forma abbreviata per scrivere un testo più articolato, il quale può contenere a sua volta altre sequenze di controllo che vengono espanse solo all'ultimo momento. Si osservi l'esempio seguente:

```
\def\resistenza{\valore{} ohm \tolleranza{} \%}
\def\valore{100}
\def\tolleranza{5}
25 resistenze \resistenza \par
\def\valore{300}
\def\tolleranza{1}
30 resistenze \resistenza \par
\bye
```

Si ottiene un testo simile a quello seguente, con cui si comprende il fatto che le parole di controllo '**\valore**' e '**\tolleranza**' vengono espanse per ultime:

25 resistenze 100 ohm 5 %
30 resistenze 300 ohm 1 %

L'espansione di una parola di controllo del genere avviene in modo letterale, salvo naturalmente l'espansione successiva del suo contenuto, ma questo fatto significa che vengono riprodotte anche le interruzioni delle righe e gli spazi. Alle volte si preferisce strutturare il contenuto di una dichiarazione del genere, per cui si utilizzano dei commenti per evitare di dare un significato agli spazi che si inseriscono. Si osservi l'esempio precedente dopo una piccola modifica:

```
\def\resistenza{

\valore{} ohm \tolleranza{} \%}

\def\valore{100}
\def\tolleranza{5}
25 resistenze \resistenza \par
\def\valore{300}
\def\tolleranza{1}
30 resistenze \resistenza \par
\bye
```

In questo caso, ciò che si ottiene è diverso, perché la parola di controllo '**\resistenza**' si espande inserendo inizialmente una riga vuota, ovvero ciò che poi si traduce nell'inizio di un paragrafo nuovo:

25 resistenze
100 ohm 5 %
30 resistenze
300 ohm 1 %

Per evitare questo tipo di inconveniente, si può mettere un commento all'inizio della riga vuota, che così perde questa sua particolarità:

```
\def\resistenza{
%
\valore{} ohm \tolleranza{} \%}
```

Di solito, in queste situazioni si mette un commento anche dopo la parentesi graffa aperta:

```
\def\resistenza{%
%
\valore{} ohm \tolleranza{} \%}
```

Oltre a questo è da tenere in considerazione che nel momento dell'espansione, ciò che si ottiene non risulta contenuto in una zona separata; in altri termini, ciò che viene dichiarato o modificato all'interno di questa definizione, continua a valere anche al di fuori. Eventualmente, se si intende che l'espansione debba generare una zona isolata, vanno usate le parentesi graffe come già mostrato. Per esempio:

```
\def\attenzione{{\bf ATTENZIONE!!!}}
```

In questo caso, l'istruzione '**\bf**' inizia la scrittura in neretto; così, la parola di controllo '**\attenzione**' permette di inserire la scritta che si vede, in neretto, senza interferire con il testo successivo.

Gli esempi usati fino a questo punto sono riconducibili all'idea di una funzione che non prevede parametri per la chiamata, ricevendo i dati attraverso variabili globali. Per dichiarare una macroistruzione in grado di ricevere degli argomenti si usa una dichiarazione come quella seguente:

```
\def\nome#1[#2...[#9]]{espressione_stringa}
```

In una dichiarazione del genere si possono indicare un massimo di nove parametri (parametri formali, a cui si abbinano nella chiamata gli argomenti attuali), rappresentati da '**#1**', '**#2**',... '**#9**', che possono essere inseriti nella stringa contenuta tra le parentesi graffe. Volendo modificare l'esempio già visto, le cose si potrebbero semplificare nel modo seguente:

```
\def\resistenza#1#2{#1 ohm #2 \%}
25 resistenze \resistenza{100}{5} \par
30 resistenze \resistenza{300}{1}
\bye
```

Nella dichiarazione, i simboli '**#*n***' che si inseriscono all'interno del testo contenuto tra parentesi graffe possono essere indicati an-

che più volte, ottenendo sempre l'espansione del parametro ***n***-esimo corrispondente.

È ormai evidente il modo in cui deve essere usata la parola di controllo che può essere chiamata con dei parametri:

```
\nome{parametro_1}[{parametro_2}[...{parametro_n}]]
```

È importante sottolineare che, contrariamente a quanto si potrebbe immaginare, la stringa utilizzata come argomento della chiamata di una macroistruzione non può essere separata in paragrafi.

Si osservi che molte macroistruzioni predefinite di TeX isolano l'espressione stringa che restituiscono all'interno di un raggruppamento, in modo tale che ciò che si cambia al suo interno non si rifletta nel testo successivo. Di solito questo fatto è un comportamento «logico», o intuitivo, ma non si deve pensare che la definizione di una macro implichi automaticamente questa forma di isolamento. In pratica, il fatto di usare una parola di controllo del tipo '***nome*{*espressione*}**', non significa implicitamente che quanto inserito come argomento della chiamata non debba anche influenzare il testo successivo.

49.4.1 Chiamata di macroistruzioni che richiedono l'indicazione di argomenti

È stata mostrata la sintassi per la chiamata di una macroistruzione che richiede di fornire uno o più argomenti:

```
\nome{parametro_1}[{parametro_2}[...{parametro_n}]]
```

Tuttavia, si tratta di una semplificazione. Infatti, gli argomenti possono anche non essere racchiusi tra parentesi graffe, ma in tal caso, il primo argomento diventa il primo carattere che segue. Si osservi l'esempio seguente, in cui si dichiara una macro con tre parametri e poi la si chiama senza raggruppare gli argomenti:

```
\def\ciao#1#2#3{%
    Primo argomento della chiamata: #1\par
    Secondo argomento della chiamata: #2\par
    Terzo argomento della chiamata: #3\par
}

\ciao, come stai?
```

Primo argomento della chiamata: ,
Secondo argomento della chiamata: c
Terzo argomento della chiamata: o
me stai?

Nella sezione 49.10 viene mostrato l'uso di macro come '**\'**' che servono ad aggiungere un accento alla lettera successiva. Queste si usano solitamente senza circoscrivere la lettera che segue entro parentesi graffe.

49.5 Riferimenti ad altre parole o simboli di controllo

È possibile creare dei riferimenti a una parola o a un simbolo di controllo, usando la sintassi seguente:

```
\let\nome_nuovo=\nome
```

In pratica, in questo modo si crea una parola di controllo alternativa a un'altra già esistente: '***nome_nuovo*** può essere usato al posto di '***nome***'.[4] Si osservi l'esempio:

```
\newdimen\altezza
\altezza=10cm
\let\grandezza=\altezza
\altezza=20cm
La scatola ha una grandezza di \the\grandezza.
\bye
```

Si ottiene in pratica il testo

La scatola ha una grandezza di 569.05511pt.

dove 569,055 11 punti corrispondono esattamente a 20 cm, ovvero l'ultimo valore assegnato alla lunghezza '**\altezza**', a cui punta anche '**\grandezza**'.

Tuttavia, se si fa un esperimento simile con una parola di controllo corrispondente a una macroistruzione definita con l'istruzione '**\def**', il riferimento che si genera con l'istruzione '**\let**' è quello che punta alla macroistruzione di quel momento, mentre una ridefinizione della parola di controllo di partenza non si riflette nel riferimento:

```
\def\resistenza#1#2{#1 ohm #2 \%}
\let\prova=\resistenza
25 resistenze \resistenza{100}{5}

25 resistenze \prova{100}{5}

\def\resistenza#1#2{#2 ohm #1 \%}
30 resistenze \resistenza{300}{1}

30 resistenze \prova{300}{1}
\bye
```

Quello che si ottiene dalla composizione di questo esempio è il testo seguente:

25 resistenze 100 ohm 5 %
25 resistenze 100 ohm 5 %
30 resistenze 1 ohm 300 %
30 resistenze 300 ohm 1 %

49.6 Testo normale e ambienti matematici

Una caratteristica molto importante di TeX è la distinzione tra due modalità di funzionamento. In pratica, si distingue un contesto «normale» da un contesto matematico. L'ambiente matematico si introduce e si conclude con il simbolo '**$**' e in tale situazione diventano disponibili delle istruzioni che non si possono utilizzare al di fuori di questo ambito, mentre alcune istruzioni dell'ambiente normale non lo sono più.

Per esempio, alcuni caratteri esistono solo nell'ambiente matematico; è già stato visto in che modo vanno indicate le parentesi graffe quando si scrive in un ambito normale, attraverso le istruzioni '**$\{$**' e '**$\}$**'. In pratica, il dollaro iniziale e finale di queste istruzioni serve ad aprire e a chiudere l'ambiente matematico.

49.7 Modalità orizzontale e modalità verticale

Nel suo lavoro di composizione, TeX distingue due situazioni, definite come ***modalità orizzontale*** e ***modalità verticale***. Per comprendere la differenza tra queste due situazioni, occorre pensare alla struttura di ciò che si inserisce in una pagina.

Ogni oggetto che viene inserito in una pagina è una scatola, con le sue dimensioni. Queste scatole si inseriscono a loro volta all'interno di altre; per esempio, una lettera è una scatola che si inserisce in una riga, ovvero un'altra scatola, che a sua volta si inserisce in un paragrafo, ovvero ancora un'altra scatola, ecc.

In base al contesto, orizzontale o verticale, TeX si occupa di inserire spazi orizzontali o verticali: tra le lettere di una parola, tra le parole, tra le righe, tra i paragrafi, ecc.

Quello che conta comprendere di tutto questo è che alcune istruzioni possono essere inserite solo in modalità orizzontale, altre solo in modalità verticale. Volendo sperimentare se un certo contesto si trovi in modalità orizzontale o verticale, si può realizzare la macro seguente e collocarla nel testo dove si ritiene opportuno; nella composizione finale si ottiene alternativamente la frase «modalità orizzontale», oppure «modalità verticale»:

```
\def\verificamodo{%
\ifvmode
    modalit\'a verticale
\else
    modalit\'a orizzontale
\fi}
```

Nella sezione 49.8 viene spiegato l'uso dell'istruzione '**\ifvmode**'.

49.8 Strutture di controllo

Il linguaggio di TeX possiede una serie di strutture di controllo condizionali, in cui parte della condizione è implicita nel nome dell'istruzione con cui la si introduce. Questo insieme di strutture ha una sintassi comune riconducibile alla semplificazione seguente:

```
\if nome_condizione [argomento]
    testo_e_altre_istruzioni_se_vero
\else
    testo_e_altre_istruzioni_se_falso
\fi
```

Spesso, per completare la struttura anche quando una delle due ipotesi non deve generare alcun risultato, si utilizza l'istruzione '**\relax**', la quale rappresenta proprio l'operazione nulla.

Nelle sezioni seguenti vengono descritte solo le strutture condizionali più comuni.

49.8.1 Istruzione «\ifodd»

È possibile verificare se un numero intero è dispari o pari con la struttura seguente:

```
\ifodd n
    testo_e_altre_istruzioni_se_dispari
\else
    testo_e_altre_istruzioni_se_pari
\fi
```

Di solito, al posto del numero ***n*** si inserisce l'istruzione '**\pageno**' che restituisce il numero della pagina corrente, permettendo così di verificare se la pagina è dispari o pari:

```
\ifodd \pageno
    testo_e_altre_istruzioni_se_pagina_dispari
\else
    testo_e_altre_istruzioni_se_pagina_pari
\fi
```

49.8.2 Istruzione «\ifmmode»

È possibile verificare se ci si trova in modalità normale o in modalità matematica con la struttura seguente:

```
\ifmmode
    testo_e_altre_istruzioni_se_modalità_matematica
\else
    testo_e_altre_istruzioni_se_modalità_normale
\fi
```

49.8.3 Istruzione «\ifnum»

È possibile mettere a confronto due numeri con la struttura seguente:

```
\ifnum m=|>|<n
    testo_e_altre_istruzioni_se_confronto_valido
\else
    testo_e_altre_istruzioni_se_confronto_non_valido
\fi
```

In pratica, il confronto può avvenire solo con gli operatori '**=**', '**>**' e '**<**', per indicare rispettivamente se i due valori sono uguali, se il primo è maggiore del secondo oppure se il primo è minore del secondo.

49.8.4 Istruzioni «\ifhmode» e «\ifvmode»

È possibile verificare se la composizione si trova in modalità orizzontale o verticale con una delle due strutture seguenti:

```
\ifhmode
    testo_e_altre_istruzioni_se_modalità_orizzontale
\else
    testo_e_altre_istruzioni_se_modalità_verticale
\fi
```

```
\ifvmode
    testo_e_altre_istruzioni_se_modalità_verticale
\else
    testo_e_altre_istruzioni_se_modalità_orizzontale
\fi
```

49.9 Verifica del significato di un'istruzione elementare

Per TeX, un'istruzione elementare può essere il singolo carattere di una parola, oppure una sequenza di controllo. Volendo comprendere il senso di qualcosa, si può verificare come intenda TeX questa o quell'istruzione. Per questo, occorre avviare la composizione indicando un file vuoto; per esempio, in un sistema Unix si potrebbe fare così:

```
$ tex /dev/null [Invio]

  This is TeX, Version 3.14159 (Web2C 7.3.1)
  (/dev/null)
  *
```

A questo punto, dall'invito, si può usare l'istruzione '**\show**' nel modo seguente:

```
\show istruzione_elementare
```

Per esempio, ci si può domandare se il carattere '**@**' abbia un significato particolare:

```
*\show @ [Invio]

  > the character @.
  <*> \show @

  ?
```

A questo punto, la composizione si ferma in attesa di altre indicazioni, mostrando un invito differente. Questo permette di comprendere anche che non conviene usare '**\show**' in un file normale del quale si vuole ottenere la composizione, perché in quel punto ci sarebbe una sospensione con richiesta di intervento. A ogni modo, per proseguire basta premere [*Invio*], quindi si può chiedere di conoscere qualcosa di diverso:

```
?[Invio]

*\show \TeX [Invio]
```

In questo caso si vuole conoscere in cosa consiste la macro '**\TeX**' ed ecco il risultato che si ottiene:

```
> \TeX=macro:
->T\kern -.1667em\lower .5ex\hbox {E}\kern -.125emX.
<*> \show \TeX

?
```

Pertanto, questo significa che '**\TeX**' è definita come:

```
\def\TeX{T\kern -.1667em\lower .5ex\hbox {E}\kern -.125emX}
```

Per completare l'esempio, conviene vedere come appare una macro che prevede l'indicazione di un parametro, come nel caso di '**\'**', che serve a mettere un accento grave su una lettera:

```
?[Invio]
```

*\show \‘ [*Invio*]

```
> \‘=macro:
#1->{\accent 18 #1}.
<*> \show \‘

?
```

In pratica, la definizione originale è la seguente:

```
\def\‘#1{{\accent 18 #1}}
```

Per terminare l’uso interattivo basta inserire alla fine l’istruzione ‘**\bye**’:

?[*Invio*]

*\bye [*Invio*]

```
No pages of output.
Transcript written on null.log.
```

49.10 Caratteri

In condizioni normali, il testo scritto nel sorgente viene riprodotto nello stesso modo nella composizione finale, dopo essere stato reimpaginato opportunamente. Tuttavia, alcuni simboli hanno significati speciali, per cui si devono usare simboli di controllo al loro posto, inoltre le lettere accentate e altri caratteri che non fanno parte dell’ASCII standard non possono essere usati direttamente, per cui servono altri simboli o parole di controllo per generarli.

A fianco di questo problema, nasce poi l’esigenza di poter scrivere utilizzando stili, forme e corpi differenti.

49.10.1 Caratteri speciali

Come è descritto nella tabella 49.34, alcuni caratteri hanno un significato speciale e non producono il simbolo corrispondente in fase di composizione. Per porre rimedio a questo problema e alla mancanza di altri caratteri, si usano generalmente delle sequenze di controllo. Tuttavia, dal momento che si distinguono due contesti di composizione differenti (quello normale e quello matematico), certi caratteri o certe sequenze di controllo esistono solo nella modalità matematica.

In generale, salvo altra indicazione, si fa sempre riferimento alla modalità di composizione normale, per cui un carattere o una sequenza di controllo che può apparire solo in modalità matematica, viene mostrata generalmente delimitandola tra due simboli ‘**$**’. Per esempio, quando si afferma che le parentesi graffe si ottengono con le istruzioni ‘**$\{$**’ e ‘**$\}$**’, è chiaro che in modalità matematica non serve più la delimitazione con i simboli ‘**$**’.

La tabella 49.88 riepiloga le sequenze di controllo per i caratteri speciali disponibili.[5]

Tabella 49.88. Elenco delle sequenze di controllo necessarie a ottenere i caratteri speciali che non richiedono la sovrapposizione di accenti.

Simbolo	Codice	Simbolo	Codice
\	\$backslash$	{	$\{$
}	$\}$	$	\$
&	\&	#	\#
^	\^{}	_	_
%	\%	~	\~{}
<	$<$	>	$>$
œ	\oe	Œ	\OE
æ	\ae	Æ	\AE
å	\aa	Å	\AA
ø	\o	Ø	\O
ł	\l	Ł	\L
ß	\ss	ı	\i
ȷ	\j	¡	!`
¿	?`	†	\dag
‡	\ddag	§	\S
¶	\P	£	{\it\$}
©	\copyright	…	\dots
TEX	\TeX		

49.10.2 Accenti

TeX ha la capacità di collocare un accento sopra ogni simbolo, anche se ciò che si ottiene può non avere senso per qualunque lingua. Questa apposizione di accenti si ottiene con sequenze di controllo che precedono la lettera da accentare. Quando si tratta di simboli di controllo, la lettera successiva può essere unita alla sequenza, mentre quando si tratta di parole di controllo occorre lasciare uno spazio di sicurezza. Per esempio, si scrive ‘**\‘o**’ per ottenere la lettera «ò», mentre si scrive ‘**\c c**’ per ottenere la lettera ‘**ç**’.[6] La tabella 49.89 riepiloga le sequenze di controllo necessarie a ottenere tutti gli accenti disponibili, usando la lettera «o» come esempio.

Tabella 49.89. Elenco delle sequenze di controllo necessarie a ottenere le lettere accentate.

Simbolo	Codice	Simbolo	Codice
ò	\`o	ó	\´o
ô	\^o	ö	\"o
õ	\~o	ō	\=o
ȯ	\.o	ŏ	\u o
ǒ	\v o	ő	\H o
o͡o	\t oo	o̧	\c o
o̱	\d o	o̲	\b o

Il fatto che TeX aggiunga l’accento sopra un simbolo, significa che occorre poi provvedere a eliminare il puntino sopra le lettere «i» e «j» minuscole. Pertanto, per scrivere la lettera «ì», occorre usare l’istruzione ‘**\‘\i**’.

49.10.3 Apici, trattini e legati automatici

TeX ha una gestione estetica particolare per alcuni caratteri. Purtroppo mette a disposizione solo virgolette alte, che vanno usate opportunamente a coppia, nella forma ‘***‘testo’***’ e ‘***‘‘testo’’***’. Si osservi l’esempio seguente e in che modo TeX lo traduce tipograficamente:

```
"virgolette doppie normali" \par
‘virgolette singole, aperte e chiuse’ \par
‘‘virgolette doppie, aperte e chiuse’’ \par
```

”virgolette doppie normali”
‘virgolette singole, aperte e chiuse’
“virgolette doppie, aperte e chiuse”

Evidentemente, è bene non usare le virgolette doppie normali (‘**"**’), a meno che ci sia una ragione precisa per farlo.

TeX è in grado di gestire tre lunghezze differenti per il trattino o la lineetta orizzontale.[7] Questi si ottengono con il trattino normale,

ovvero il segno meno, usato ripetutamente. In pratica, '**-**' genera un trattino corto; '**--**' genera un trattino medio e '**---**' genera un trattino lungo. Questo tipo di automatismo può creare dei problemi quando il trattino deve assumere proprio il significato di un segno meno e non è molto bella l'idea di usare due trattini per ottenere una lineetta media. In questo caso, si usa preferibilmente la sequenza '**$-$**', dal momento che nella modalità matematica questo comportamento non ha luogo. Si osservi l'esempio:

```
Trattino corto: - \par
Trattino medio: -- \par
Trattino lungo: --- \par
Segno meno: $-$ \par
```

Trattino corto: -
Trattino medio: –
Trattino lungo: —
Segno meno: −

Per motivi estetici, la tradizione tipografica richiede che alcuni accoppiamenti di lettere vengano legati assieme. Si tratta dell'unione della lettera «f» con «i», «l» e «f». TeX rispetta questa tradizione, ma solo nella modalità normale, perché in modalità matematica le esigenze sono differenti. Si osservi l'esempio:

```
ff fi fl ffi ffl \par
$ff fi fl ffi ffl$ \par
```

ff fi fl ffi ffl
$ff\,fi\,fl\,ffi\,ffl$

49.10.4 Istruzioni alternative per generare simboli particolari

Alcune tastiere non consentono di utilizzare facilmente dei simboli che per TeX sono invece importanti. In particolare, ci possono essere difficoltà a ottenere un apostrofo corrispondente all'accento grave. Quando l'intento è solo quello di scrivere degli apici, singoli o doppi, si possono usare le parole di controllo '**\lq**' e '**\rq**' per aprire e chiudere rispettivamente. Per esempio,

```
\lq\lq ciao\rq\rq
```

racchiude tra apici doppi la parola «ciao». Tuttavia, questo sistema non può sostituire i simboli di controllo usati per ottenere l'accento grave e acuto. Per ovviare all'inconveniente, si può leggere il codice delle macroistruzioni standard di TeX per scopre un metodo alternativo, oppure si può usare l'istruzione '**\show**' in modalità interattiva:

```
\def\‘#1{{\accent18 #1}}
\def\’#1{{\accent19 #1}}
\def\v#1{{\accent20 #1}}
\def\u#1{{\accent21 #1}}
\def\=#1{{\accent22 #1}}
\def\^#1{{\accent94 #1}}
\def\.#1{{\accent95 #1}}
\def\H#1{{\accent"7D #1}}
\def\~#1{{\accent"7E #1}}
\def\"#1{{\accent"7F #1}}
```

In pratica, è sufficiente definire una parola di controllo alternativa alle sequenze che non si possono riprodurre facilmente. Ecco un esempio concreto per gli accenti grave a acuto:

```
\def\graveaccent#1{{\accent18 #1}}
\def\acuteaccent#1{{\accent19 #1}}
```

In questo modo, al posto di '**\‘e**' si può scrivere '**\graveaccent{e}**', anche se ciò richiede una digitazione un po' lunga.

Per quanto riguarda le parentesi quadre, queste possono essere rappresentate con la coppia '**\lbrack**' e '**\rbrack**' per rappresentare rispettivamente la parentesi aperta e quella chiusa. Per le parentesi graffe le cose si complicano: per inserirle nel testo come simboli tipografici, si possono usare le istruzioni '**$\lbrace$**' e '**$\rbrace$**', che come si vede richiedono un contesto matematico, come già avveniva con '**$\{$**' e '**$\}$**', ma questo significa anche che non c'è un'alternativa alle parentesi graffe usate nelle istruzioni con finalità differenti, salvo la possibilità di usare '**\bgroup**' e '**\egroup**' quando si tratta di raggruppamenti.

Tabella 49.99. Alternative ad alcune sequenze di controllo che possono creare difficoltà con tastiere incomplete.

Sequenza normale	Alternativa
‘	\lq
’	\rq
\‘	{\accent 18 *x*}
\’	{\accent 19 *x*}
[	\lbrack
]	\rbrack
$\{$	$\lbrace$
$\}$	$\rbrace$

49.10.5 Caratteri da stampa standard

In condizioni normali, TeX mette a disposizione un carattere tondo con grazie. È possibile cambiare il gruppo stilistico e la forma del carattere con comandi generici, che non fanno riferimento a un nome preciso, ma a un aspetto di massima. Questi comandi hanno effetto dal momento in cui vengono inseriti, fino a quando ne viene incontrato un altro, oppure fino alla fine del raggruppamento in cui si trovano. Si tratta di parole di controllo di due sole lettere, elencate brevemente nella tabella 49.100.

Tabella 49.100. Selezione standard del tipo di carattere.

Sequenza	Nome	Significato	Nome corrispondente
\rm	*roman*	tondo	cmr10
\bf	*boldface*	neretto	cmbx10
\it	*italic*	corsivo	cmti10
\sl	*slanted*	inclinato	cmsl10
\tt	*typewriter*	dattilografico	cmtt10

È importante osservare che l'uso di queste parole di controllo non va a sommarsi con le precedenti, cambiando il gruppo stilistico e la forma simultaneamente. In pratica, in condizioni normali, non si può ottenere un neretto-corsivo o un neretto-inclinato combinando i comandi relativi. Si osservi l'esempio che segue:

```
Testo iniziale, \bf testo in neretto, \it testo in corsivo,
\sl testo inclinato, \tt testo dattilografico, \rm testo
tondo normale.
L'uso delle parentesi graffe {\it consente di circoscrivere}
l'effetto dei comandi che intervengono sul tipo di
carattere.
```

Testo iniziale, **testo in neretto,** *testo in corsivo, testo inclinato,* `testo dattilografico,` testo tondo normale. L'uso delle parentesi graffe *consente di circoscrivere* l'effetto dei comandi che intervengono sul tipo di carattere.

Prima di proseguire sull'argomento è il caso di osservare un problema che si presenta nel momento in cui si utilizza un carattere inclinato o corsivo: lo spostamento verso destra può creare degli inconvenienti, quando si incontra con un testo non inclinato. In caso di necessità si può aggiungere un piccolo spazio, inteso come correzione del corsivo, con il simbolo di controllo '**\/**', che rappresenta simbolicamente il concetto. Si osservi l'esempio seguente:

```
(testo normale tra parentesi tonde che finisce con una lettera l) \par
({\it testo corsivo tra parentesi tonde che finisce con una lettera l}) \par
({\it testo corsivo con correzione dopo la lettera l}\/) \par
```

(testo normale tra parentesi tonde che finisce con una lettera l)
(*testo corsivo tra parentesi tonde che finisce con una lettera l*)
(*testo corsivo con correzione dopo la lettera l*)

Le parole di controllo '**\rm**', '**\it**', '**\sl**', '**\bf**' e '**\tt**' sono delle piccole macroistruzioni standard di TeX. Infatti, le informazioni sui caratteri da stampa sono contenute in file particolari che possono essere utilizzati con l'istruzione seguente:

```
\font\nome=carattere
```

In pratica, in questo modo si dichiara una parola di controllo a cui si attribuisce il carattere indicato a destra. Il nome del carattere corrisponde solitamente alla prima parte del nome del file che lo contiene (la radice che rimane togliendo l'estensione) e si può esprimere usando indifferentemente lettere maiuscole o minuscole. Nell'esempio seguente, si dichiara un carattere tondo senza grazie, corrispondente al nome '**cmss10**':

```
\font\sans=cmss10
\sans Ecco un testo senza grazie, scritto con il carattere
''Computer Modern Sans Serif'' da 10 punti.
```

Ecco un testo senza grazie, scritto con il carattere "Computer Modern Sans Serif" da 10 punti.

Il carattere da stampa che si seleziona in questo modo ha già un corpo stabilito. In questo caso si tratta di un carattere da 10 punti. Per cambiare corpo, si può cambiare carattere, oppure si può ingrandire e ridurre espressamente, intervenendo nell'istruzione di dichiarazione:

```
\font\nome=carattere scaled n
```

Il numero ***n*** che si indica alla fine esprime un rapporto in millesimi dell'adattamento che si vuole ottenere. Per esempio,

```
\font\sans=cmss10 scaled 1200
```

richiede espressamente di abbinare alla parola di controllo '**\sans**' il carattere '**cmss10**' al 120 %, cioè ingrandito di un quinto in più.

```
\font\sans=cmss10 scaled 1200
\sans Ecco un testo senza grazie, scritto con il carattere
''Computer Modern Sans Serif'' da 10 punti, ingrandito
al 120 \%.
```

Ecco un testo senza grazie, scritto con il carattere "Computer Modern Sans Serif" da 10 punti, ingrandito al 120 %.

Naturalmente, con lo stesso criterio si può anche ridurre un carattere, sempre tenendo presente che 1000 rappresenta la dimensione di partenza. Tuttavia, in generale non si usa questa forma, sostituendo il numero con un'istruzione apposita, utile a livello mnemonico:

```
\font\nome=carattere scaled \magstep n
```

In questo caso, il numero ***n*** può andare da zero a cinque, dove zero corrisponde al corpo attuale, mentre valori maggiori rappresentano piccoli incrementi percentuali. A ogni modo si può verificare facilmente cosa succede, con il codice TeX seguente:

```
\newcount\ingrandimento
\ingrandimento=\magstep 1
\font\sans=cmss10 scaled \magstep 1
\sans Ecco un testo senza grazie, scritto con il carattere
''Computer Modern Sans Serif'' da 10 punti, ingrandito al
\the\ingrandimento{} per mille.
```

Ecco un testo senza grazie, scritto con il carattere "Computer Modern Sans Serif" da 10 punti, ingrandito al 1200 per mille.

A questo punto si pone il problema di comprendere l'utilità di usare un carattere da stampa adatto alla dimensione del corpo che interessa. In pratica, se il carattere corrispondente alla sigla '**cmr5**' è fatto per un corpo da cinque punti, quando serve un corpo da 12 è bene usare il carattere '**cmr12**', se disponibile, invece di ingrandire il primo 2,4 volte (240 %). Si osservi l'esempio seguente, in cui si mettono a confronto diversi caratteri di piccole dimensioni che vengono ingranditi tutti in modo da avere un corpo finale di 12 punti:

```
\font\cinque=cmr5 scaled 2400
\cinque Il carattere cmr5 ingrandito al 240 \% \par
\font\sei=cmr6 scaled 2000
\sei Il carattere cmr6 ingrandito al 200 \% \par
\font\sette=cmr7 scaled 1714
\sette Il carattere cmr7 ingrandito al 171,4 \% \par
\font\otto=cmr8 scaled 1500
\otto Il carattere cmr8 ingrandito al 150 \% \par
\font\nove=cmr9 scaled 1333
\nove Il carattere cmr9 ingrandito al 133 \% \par
\font\dieci=cmr10 scaled 1200
\dieci Il carattere cmr10 ingrandito al 120 \% \par
\font\normale=cmr12
\normale Il carattere cmr12 senza ingrandimenti \par
```

Il carattere cmr5 ingrandito al 240 %
Il carattere cmr6 ingrandito al 200 %
Il carattere cmr7 ingrandito al 171,4 %
Il carattere cmr8 ingrandito al 150 %
Il carattere cmr9 ingrandito al 133 %
Il carattere cmr10 ingrandito al 120 %
Il carattere cmr12 senza ingrandimenti

Tabella 49.114. Nomi dei caratteri da stampa standard a disposizione (Computer Modern).

Sigla	Denominazione originale	Corpo
cmbsy10	Computer Modern Bold Symbols	10 punti
cmbx5 cmbx6 cmbx7 cmbx8 cmbx9 cmbx10 cmbx12	Computer Modern Bold Extended Roman	5, 6, 7, 8, 9, 10 e 12 punti
cmbxsl10	Computer Modern Bold Extended Slanted Roman	10 punti
cmcsc10	Computer Modern Caps and Small Caps	10 punti
cmdunh10	Computer Modern Dunhill Roman	10 punti
cmex9 cmex10	Computer Modern Math Extension	9 e 10 punti
cmff10	Computer Modern Funny Roman	10 punti
cmfi10	Computer Modern Funny Italic	10 punti
cmfib8	Computer Modern Fibonacci Font	8 punti
cminch	Computer Modern Inch-High Sans Serif Bold Extended Caps and Digits	1 pollice
cmitt10	Computer Modern Italic Typewriter Text	10 punti
cmmi5 cmmi6 cmmi7 cmmi8 cmmi9 cmmi10 cmmi12	Computer Modern Math Italic	5, 6, 7, 8, 9, 10 e 12 punti
cmr5 cmr6 cmr7 cmr8 cmr9 cmr10 cmr12 cmr17	Computer Modern Roman	5, 6, 7, 8, 9, 10, 12 e 17 punti
cmsl8 cmsl9 cmsl10 cmsl12	Computer Modern Slanted Roman	8, 9, 10 e 12 punti
cmsltt10	Computer Modern Slanted Typewriter Text	10 punti
cmss8 cmss9 cmss10 cmss12 cmss17	Computer Modern Sans Serif	8, 9, 10, 12 e 17 punti
cmssbx10	Computer Modern Sans Serif Bold Extended	10 punti
cmssdc10	Computer Modern Sans Serif Demibold Condensed	10 punti

Sigla	Denominazione originale	Corpo
cmssi8 cmssi9 cmssi10 cmssi12 cmssi17	Computer Modern Slanted Sans Serif	8, 9, 10, 12 e 17 punti
cmssq8	Computer Modern Sans Serif Quotation Style	8 punti
cmssqi8	Computer Modern Sans Serif Quotation Style Slanted	8 punti
cmsy5 cmsy6 cmsy7 cmsy8 cmsy9 cmsy10	Computer Modern Math Symbols	5, 6, 7, 8, 9 e 10 punti
cmtcsc10	Computer Modern Typewriter Caps and Small Caps	10 punti
cmtex8 cmtex9	Computer Modern TeX Extended ASCII characters	8 e 9 punti
cmti7 cmti8 cmti9 cmti10 cmti12	Computer Modern Text Italic	7, 8, 9, 10 e 12 punti
cmtt8 cmtt9 cmtt10 cmtt12	Computer Modern Typewriter Text	8, 9, 10 e 12 punti
cmu10	Computer Modern Unslanted Italic	10 punti
cmvtt10	Computer Modern Variable-width Typewriter Text	10 punti

A fianco delle parole di controllo generiche per la selezione del carattere da stampa, esistono altre parole di controllo standard che contengono anche l'informazione sulla dimensione. Si tratta dell'elenco riportato nella tabella 49.115. Ciò permette di non utilizzare una definizione dettagliata del carattere quando quello che serve cambiare è solo la dimensione (entro i limiti normali già previsti).

Tabella 49.115. Parole di controllo per la selezione di un carattere da stampa generico, con l'indicazione della dimensione.

Parola di controllo	Effetto
`\tenrm`	Tondo, 10 punti.
`\sevenrm`	Tondo, sette punti.
`\fiverm`	Tondo, cinque punti.
`\tenit`	Corsivo, 10 punti.
`\tenbf`	Neretto, 10 punti.
`\sevenbf`	Neretto, sette punti.
`\fivebf`	Neretto, cinque punti.
`\tentt`	Dattilografico, 10 punti.
`\tensl`	Inclinato, 10 punti.
`\teni`	Corsivo matematico, 10 punti.
`\seveni`	Corsivo matematico, sette punti.
`\fivei`	Corsivo matematico, cinque punti.
`\tensy`	Simboli matematici, 10 punti.
`\sevensy`	Simboli matematici, sette punti.
`\fivesy`	Simboli matematici, cinque punti.
`\tenex`	Estensione ai simboli matematici, 10 punti.

49.10.6 Sottolineatura

La gestione della sottolineatura con TeX è separata da quella che riguarda la forma e la dimensione dei caratteri. Si tratta solo di una linea orizzontale che viene collocata al di sotto del testo, a una distanza che varia in funzione delle caratteristiche del testo, che si ottiene con la macro '**`\underbar`**':

```
\underbar{testo_da_sottolineare}
```

La linea con cui si ottiene questa sottolineatura crea implicitamente una scatola orizzontale (sezione 49.14) che non può essere suddivisa in più righe. Si osservi l'esempio seguente:

```
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 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 \underbar{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 bla
bla.
```

Dal momento che il testo sottolineato è molto lungo e si trova verso la fine di una riga, TeX non può separarlo e si crea un problema estetico molto spiacevole:

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 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 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 bla bla bla bla <u>bla bla bla bla bla bla bla bla bla bla bla bla</u>
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.

49.11 La pagina

In condizioni normali, la pagina a cui si fa riferimento con TeX è il tipo Lettera, larga 8,5 pollici e alta 11 pollici. Nell'ambito di questo spazio, il testo occupa normalmente un rettangolo di 6,5 pollici per 8,9 pollici, con un margine uguale di un pollice (salvo il margine inferiore che è di 1,1 pollici).

Al di sopra e al di sotto di questo rettangolo è possibile collocare una riga, con lo scopo normale di indicare il nome della sezione o il numero della pagina. In condizioni normali, in basso appare proprio il numero della pagina, ma può essere fatto sparire con l'istruzione '**`\nopagenumbers`**'.

Le dimensioni della pagina possono essere modificate, ma non si fa riferimento a una pagina e ai suoi margini; si interviene piuttosto sulla posizione di partenza, sulla larghezza (ovvero la giustezza) e sull'altezza dello spazio a disposizione dei paragrafi.

49.11.1 La parte centrale della pagina

La parte centrale della pagina è controllata da quattro variabili, rappresentate dalle parole di controllo '**`\hoffset`**', '**`\voffset`**', '**`\hsize`**' e '**`\vsize`**', le quali controllano rispettivamente le coordinate iniziali, la giustezza e l'altezza dell'area di scrittura.

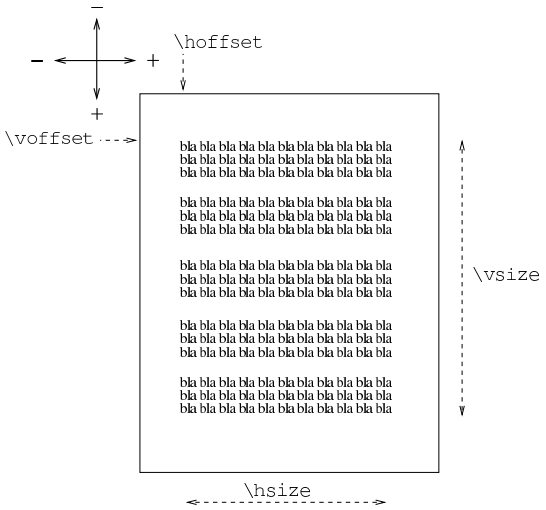
Tabella 49.118. Parole di controllo che regolano le dimensioni e la collocazione dell'area di scrittura centrale nella pagina.

Parola di controllo	Competenza	Valore predefinito
`\hoffset`	Posizione orizzontale iniziale	0
`\voffset`	Posizione verticale iniziale	0
`\hsize`	Giustezza, a partire da '**`\voffset`**'	6,5 pollici
`\vsize`	Altezza, a partire da '**`\hoffset`**'	8,9 pollici

È importante osservare che le coordinate zero di '**`\hoffset`**' e

‘**\voffset**’ rappresentano un punto in alto a sinistra del foglio, collocato a destra e in basso di un pollice rispetto all’angolo superiore sinistro del foglio stesso. In pratica, per indicare valori inferiori del margine superiore e di quello sinistro, si devono attribuire a queste parole di controllo dei valori negativi.

Figura 49.119. Area del testo nella pagina.



L’esempio seguente consente di verificare il valore predefinito che viene restituito da queste parole di controllo:

```
$\backslash$hoffset = \the\hoffset \par
$\backslash$voffset = \the\voffset \par
$\backslash$hsize = \the\hsize \par
$\backslash$vsize = \the\vsize \par
```

$\backslash\text{hoffset} = 0.0\text{pt}$
$\backslash\text{voffset} = 0.0\text{pt}$
$\backslash\text{hsize} = 398.33858\text{pt}$
$\backslash\text{vsize} = 643.20255\text{pt}$

I valori vengono mostrati in punti, ma si verifica facilmente la corrispondenza con quanto mostrato nella tabella 49.118.

È importante sottolineare che le coordinate di partenza rappresentano l’angolo superiore sinistro dell’area in cui si inserisce il testo; pertanto, l’altezza espressa dalla parola di controllo ‘**\vsize**’, si sviluppa in basso, a partire da quel punto.

Il testo che si scrive nel sorgente TeX parte dalle coordinate iniziali e, salvo l’uso di comandi appositi, viene impaginato automaticamente; quando lo spazio verticale viene esaurito, si passa automaticamente a un’altra pagina.

È importante sottolineare che le modifiche apportate alle coordinate iniziali e all’altezza del testo della pagina hanno effetto a partire dalla pagina corrente; inoltre, se i valori vengono cambiati più volte, contano le ultime modifiche che si possono ricondurre alla pagina corrente. Fa eccezione a questa regola la giustezza del testo, controllata dalla parola di controllo ‘**\hsize**’, che può essere modificata in qualunque momento, intervenendo a partire dal paragrafo in cui appare, senza coinvolgere il testo precedente nella stessa pagina.

Eventualmente, se prima di scrivere alcunché, si colloca un’istruzione come quella seguente,

```
\magnification=n
```

si ottiene un ingrandimento o una riduzione di tutte le dimensioni, compreso il corpo dei caratteri. In pratica, il numero ***n*** che si assegna a ‘**\magnification**’ è un valore che esprime la riduzione o l’ingrandimento in rapporto a 1000, come già avveniva con il ridimensionamento dei caratteri da stampa. In questo senso, al posto del numero si può usare anche la parola di controllo ‘**\magstep**’, come è già stato mostrato in precedenza:

```
\magnification=\magstep n
```

In questo caso, ***n*** esprime un valore intero, da zero a cinque.

La definizione di un ingrandimento o di una riduzione ha effetto su tutti i comandi che definiscono una dimensione; per esempio, se si usa un ingrandimento di 2000, pari al doppio, volendo indicare una dimensione di un centimetro, si ottengono in pratica due centimetri.

Quando si intende indicare una lunghezza esatta, che non possa essere ridimensionata, si usa la parola chiave ‘**true**’ davanti all’unità di misura. Per esempio, ‘**5cm**’ è una lunghezza adattabile, mentre ‘**5truecm**’, oppure ‘**5 true cm**’, indica sempre cinque centimetri.

In condizioni normali, TeX cerca di occupare tutto lo spazio orizzontale e tutto lo spazio verticale, giustificando orizzontalmente e verticalmente (ovvero allineando simultaneamente a sinistra e a destra, in alto e in basso). In particolare, l’allineamento verticale del testo viene controllato da due parole di controllo: ‘**\normalbottom**’ e ‘**\raggedbottom**’. Nel primo caso si ha la situazione «normale», ovvero l’allungamento del testo in modo da completare lo spazio verticale di ogni pagina, mentre nel secondo caso questo allungamento non ha luogo.

TeX mette a disposizione un’istruzione per il salto pagina anticipato che si ottiene con la parola di controllo ‘**\eject**’. Quando l’impaginazione è quella normale, ovvero quella corrispondente alla parola di controllo ‘**\normalbottom**’, l’inserimento di un salto pagina costringe TeX a fare in modo che il testo esistente, per quanto breve possa essere, finisca esattamente alla fine di ‘**\vsize**’. Per evitare di produrre degli allungamenti sgradevoli del testo, pur senza disabilitare globalmente la funzionalità di allineamento verticale, si può usare un comando per l’inserimento di uno spazio verticale allungabile in modo indefinito, il quale si ottiene con la parola di controllo ‘**\vfill**’. Pertanto, quando si vuole inserire un salto pagina si usa generalmente un’istruzione composta da entrambe le parole di controllo, come nell’esempio seguente:

```
\vfill \eject
```

Inizialmente è stato indicato l’uso dell’istruzione ‘**\bye**’ per concludere un sorgente TeX. In realtà, ‘**\bye**’ è una macroistruzione realizzata per concludere bene un testo, in cui si utilizza anche ‘**\vfill**’ per non allungare il contenuto della pagina stessa. L’istruzione primitiva per concludere il documento è invece ‘**\end**’.

Anche ‘**\eject**’ è una macro, che utilizza in pratica ‘**\break**’, dopo un passaggio in modalità verticale, in modo da dare a ‘**\break**’ il contesto corretto.

Eventualmente, esiste anche un’altra macro, oltre a ‘**\eject**’, il cui scopo è quello di garantire il salto pagina in ogni circostanza. Si tratta di ‘**\supereject**’.

49.11.2 Intestazione e fondo pagina

Oltre alla parte centrale della pagina, è possibile accedere a una riga di intestazione e a una riga alla base della pagina, le quali appaiono rispettivamente sullo spazio del margine superiore e del margine inferiore. Queste righe vengono annotate in due variabili apposite, a cui si accede con le parole di controllo ‘**\headline**’ e ‘**\footline**’:

```
\headline={intestazione}
```

```
\footline={fondo_pagina}
```

In condizioni normali, la variabile che contiene la riga di fondo pagina è già impostata con l'espressione '**\folio\pageno**', con la quale si estrae dal contatore '**\pageno**' il valore corrispondente al numero di pagina. Inoltre, per eliminare il fondo pagina predefinito in questo modo, si utilizza normalmente la parola di controllo '**\nopagenumbers**', come sinonimo di azzeramento, al posto di scrivere '**\footline={}**'.

Il testo che viene associato all'intestazione e al fondo pagina, risulta allineato a destra e a sinistra, simultaneamente. Tuttavia, attraverso l'uso della parola di controllo '**\hfil**', si riescono a ottenere degli effetti utili di allineamento. La parola di controllo '**\hfil**' ha la proprietà di inserire uno spazio allungabile, ma può essere utilizzata più volte, suddividendo equamente lo spazio della riga. Per esempio, scrivendo

```
\headline={sinistra \hfil centro \hfil destra}
```

si ottiene la suddivisione che si può vedere qui sotto:

sinistra centro destra

Quando si distingue tra pagine destre e sinistre, si può usare la struttura condizionale comandata dalla parola di controllo '**\ifodd**' per modificare l'aspetto delle intestazioni o dei fondo pagina. Nell'esempio seguente si vuole mettere il numero della pagina nell'intestazione, facendo in modo che appaia verso l'estremità esterna della pagina:

```
\headline={%
\ifodd \pageno
    {bla bla bla \hfil \the\pageno}
\else
    {\the\pageno \hfil bla bla bla}
\fi}
```

Nell'esempio vengono racchiuse le due alternative tra parentesi graffe, perché altrimenti '**\pageno**' creerebbe dei problemi con la prima ipotesi. Non sarebbe necessario il raggruppamento della seconda alternativa, ma qui lo si fa per motivi «estetici».

L'espressione dell'intestazione o del fondo pagina, viene espansa ogni volta, nel momento in cui serve. Tuttavia, in questo modo, le impostazioni riferite al carattere da stampa usato nel testo normale potrebbero influenzare l'aspetto di queste righe. Per sicurezza, conviene completare la definizione di intestazione e fondo pagina con l'indicazione del carattere da stampa, completo dell'informazione sulla sua dimensione, come nell'esempio seguente, in cui si richiede espressamente un carattere inclinato di 10 punti:

```
\headline={\tensl Questa \'e una bella intestazione}
```

49.11.3 Numerazione delle pagine

La numerazione delle pagine avviene in modo automatico, attraverso la gestione del contatore '**\pageno**'. In condizioni normali, la prima pagina ha il numero uno e a ogni salto pagina si ha un incremento di questo valore.

L'impostazione predefinita della riga che appare a fondo pagina, è tale per cui se il contatore contiene un valore positivo, si mostra un numero arabo normale, mentre se il valore è negativo, si mostra la pagina in numero romano. Per la precisione, ciò si ottiene attraverso la parola di controllo '**\folio**' che, senza bisogno di trasformazioni, restituisce il numero della pagina secondo questa convenzione.

In particolare, se il contatore '**\pageno**' contiene un valore positivo, ogni salto pagina continua a incrementarlo, mentre se ha un valore negativo, si ottiene un decremento, proprio in base alla logica della riga di fondo pagina predefinita.

Nel momento in cui si intende gestire il numero della pagina che appare nel fondo pagina, o nella riga di intestazione, non ha più importanza questa convenzione sul segno positivo o negativo; quello che conta è reinizializzare il contatore quando serve.

Il vero problema che si pone è quello di rappresentare le pagine attraverso numeri romani con lettere maiuscole. L'esempio seguente mostra la macro '**\Roman**' realizzata per ottenere la trasformazione in numero romano maiuscolo del suo argomento:

```
\def\Roman#1{\uppercase\expandafter{\romannumeral #1}}
```

La macro va usata ovviamente così:

```
\Roman{n}
```

49.11.4 Note a piè di pagina

Le note a piè di pagina sono gestite automaticamente attraverso la macro standard '**\footnote**', inserendole alla base della pagina, entro lo spazio stabilito da '**\vsize**'.

```
\footnote{simbolo}{testo_della_nota}
```

In pratica, la nota viene inserita nel testo, nel punto in cui si vuole che appaia il simbolo definito come primo parametro, poi la composizione colloca la nota dove più appropriato, preceduta sempre dallo stesso simbolo di riferimento.

Il simbolo usato per fare riferimento alla nota può essere scelto liberamente, tenendo conto che TeX non fa nulla per adattarlo. Pertanto, se l'intenzione è quella di usare il classico numero ad apice, occorre provvedere esplicitamente a sollevarlo, come nell'esempio seguente:

```
Bla bla\footnote{$^1$}{Questa è la mia prima nota a piè
pagina.} bla bla...
```

Generalmente, si numerano le note a piè pagina, ma TeX non offre un meccanismo automatico di numerazione. Tuttavia è facile rimediare attraverso la definizione di un contatore apposito:

```
\newcount\nota
\nota=0
\def\annotazione#1{\advance\nota by 1
                   \footnote{$^{\number\nota}$}{#1}}
```

Successivamente, invece di usare direttamente '**\footnote**', si può inserire la macro '**\annotazione**', la quale provvede da sola a incrementare il contatore '**\nota**', inserendo il numero corrispondente come apice. Eventualmente, quando si comincia con una sezione importante, che parte da una pagina nuova, può essere conveniente riazzerare il conteggio.

L'esempio mostrato però non è completo. Infatti, la macro '**\annotazione**' potrebbe essere inserita per qualche ragione in un gruppo isolato, per cui, l'incremento del contatore della pagina verrebbe perduto; inoltre, il tipo di carattere usato per le note viene ereditato dal punto in cui si trova l'inserimento. Per ovviare a questi inconvenienti, si può modificare la macro '**\annotazione**' nel modo seguente:

```
\def\annotazione#1{\global\advance\nota by 1
\footnote{$^{\number\nota}$}{\sevenrm #1}}
```

In questo modo, l'incremento del contatore diventa globale e si fissa il carattere da usare per le note in un tondo normale con un corpo da sette punti.

49.11.5 Oggetti fluttuanti

Nell'ambito del testo è possibile individuare delle zone che devono rimanere unite, nel senso che non possono apparire separate tra una pagina e la successiva, per cui si stabilisce una collocazione che non segue il flusso normale del testo. Questi ambienti sono introdotti da una parola di controllo particolare:

```
\midinsert
testo_e_altre_istruzioni
...
\endinsert
```

```
\topinsert
testo_e_altre_istruzioni
...
\endinsert
```

```
\pageinsert
testo_e_altre_istruzioni
...
\endinsert
```

Nel primo caso, quanto contenuto tra '**\midinsert**' e '**\endinsert**' viene collocato nella posizione in cui appare, se c'è lo spazio verticale sufficiente nella pagina, altrimenti viene spostato all'inizio della pagina successiva, lasciando che il testo che seguirebbe venga anticipato al suo posto nella pagina che termina.

Nel secondo caso, quanto contenuto tra '**\topinsert**' e '**\endinsert**' viene collocato all'inizio della pagina in cui si trova, anticipando eventualmente del testo che lo precede.

Nel terzo caso, quanto contenuto tra '**\pageinsert**' e '**\endinsert**' viene collocato in una pagina indipendente, che non può contenere altro.

Questi ambienti, eventualmente con l'uso di spazi verticali, permettono comunemente di riservare degli spazi per le figure.

49.12 Paragrafi

«

Il testo scritto con TeX viene reimpaginato in paragrafi, corrispondenti ai blocchi separati da almeno una riga bianca o vuota, oppure dalla parola di controllo '**\par**'.

Le caratteristiche comuni dei paragrafi vengono definite dai valori contenuti in alcune variabili, a cui si accede attraverso altrettante parole di controllo. Si tratta della giustezza, corrispondente a '**\hsize**', della dimensione del rientro della prima riga, corrispondente a '**\parindent**', della distanza tra le righe (tra la base delle righe), corrispondente a '**\baselineskip**', della distanza tra i paragrafi, corrispondente a '**\parskip**'.

Tabella 49.131. Parole di controllo che regolano o condizionano le caratteristiche dei paragrafi comuni.

Parola di controllo	Competenza	Condizione o valore predefinito
`\hoffset`	Posizione iniziale dei paragrafi nella pagina.	0
`\hsize`	Larghezza del paragrafo a partire da '**\hoffset**'.	6,5 pollici
`\parindent`	Rientro della prima riga.	20 punti
`\baselineskip`	Distanza tra la base di una riga e la base della riga successiva.	12 punti
`\parskip`	Distanza aggiuntiva tra i paragrafi.	0
`\raggedright`	Allinea il testo a sinistra.	allineato simultaneamente a sinistra e a destra
`\leftskip`	Rientro sinistro complessivo.	0
`\rightskip`	Rientro destro complessivo.	0

Si può provare a verificare i valori predefiniti con l'esempio seguente:

```
hoffset = \the\hoffset \par
hsize = \the\hsize \par
parindent = \the\parindent \par
baselineskip = \the\baselineskip \par
parskip = \the\parskip \par
leftskip = \the\leftskip \par
rightskip = \the\rightskip \par
```


hoffset = 0.0pt
hsize = 341.43306pt
parindent = 20.0pt
baselineskip = 12.0pt
parskip = 0.0pt plus 1.0pt
leftskip = 0.0pt
rightskip = 0.0pt

Sia '**\parskip**' che '**\baselineskip**' consentono l'attribuzione di lunghezze elastiche. Come si può osservare dall'esempio, il valore predefinito di '**\baselineskip**' non prevede elasticità, mentre '**\parskip**' consente un leggero allungamento. Si tenga presente che, anche se è stato indicato un allungamento di un solo punto, in pratica, nella composizione finale si possono ottenere allungamenti molto maggiori in caso di necessità.

Come si intende, in condizioni normali i paragrafi iniziano con un rientro, senza essere distanziati tra di loro. Se questo tipo di approccio è ciò che si desidera, quando eccezionalmente si vuole un paragrafo senza rientro, come si fa di solito dopo un titolo, basta usare la parola di controllo '**\noindent**' che ha effetto solo nel paragrafo in cui si trova.

Per modificare la posizione verticale e la larghezza dei paragrafi, invece di intervenire sul valore controllato dalla parola di controllo '**\hsize**' (su '**\hoffset**' non si può intervenire, perché riguarda la pagina intera), conviene usare invece '**\leftskip**' e '**\rightskip**', le quali si riferiscono rispettivamente a un rientro sinistro e a un rientro destro. In pratica, se invece si vuole estendere il paragrafo oltre il margine sinistro fissato da '**\hoffset**', oppure si vuole andare oltre il margine destro stabilito da '**\hsize**', basta usare valori negativi per i rientri. L'esempio seguente mostra il caso di un paragrafo rientrato a sinistra e a destra per scrivere un'annotazione che risalti agli occhi del lettore:

```
\leftskip=1cm
\rightskip=1cm
{\bf Attenzione}: 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 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.

\leftskip=0cm
\rightskip=0cm
Questo \'e invece un paragrafo normale 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.
```

Attenzione: 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 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.

Questo è invece un paragrafo normale 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 bla bla.

Si vede dall'esempio la necessità di intervenire successivamente per ripristinare i valori normali dei rientri.

I rientri definiti attraverso '**\leftskip**' e '**\rightskip**', possono essere gestiti anche attraverso la macroistruzione '**\narrower**', con cui si incrementa il loro valore di '**\parskip**'. In pratica, usando l'istruzione '**\narrower**' i margini del testo vengono aumentati di quanto contenuto in '**\parskip**'. Naturalmente, l'effetto di tale istruzione vale per tutti i paragrafi successivi, a meno che sia inserita in un raggruppamento; in alternativa si può intervenire nuovamente su '**\leftskip**' e '**\rightskip**' per ripristinare dei valori normali.

Un paragrafo risulta normalmente allineato simultaneamente a sinistra e a destra. Se si vuole eliminare l'allineamento destro, si utilizza la parola di controllo '**\raggedright**':

```
Questo \'e un paragrafo normale allineato a sinistra e a
destra, ma non tutti gli autori gradiscono questo sistema di
impaginazione orizzontale. Bla bla bla bla bla bla bla bla
```

```
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla.

{\raggedright In questo paragrafo, invece, l'allineamento
del testo al margine destro, non ha luogo. Purtroppo, non
c'\'e una parola di controllo opposta per ripristinare
l'allineamento normale, pertanto, occorre racchiudere
l'istruzione in un raggruppamento. Si osservi che il
paragrafo viene concluso prima della fine del
raggruppamento. Bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla. \par}

Questo \'e un altro paragrafo normale allineato a sinistra e
a destra. 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.
```

Questo è un paragrafo normale allineato a sinistra e a destra, ma non tutti gli autori gradiscono questo sistema di impaginazione orizzontale. 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.

In questo paragrafo, invece, l'allineamento del testo al margine destro,
non ha luogo. Purtroppo, non c'è una parola di controllo opposta per ripristinare
l'allineamento normale, pertanto, occorre racchiudere l'istruzione in un rag-
gruppamento. Si osservi che il paragrafo viene concluso prima della fine del
raggruppamento. Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla.

Questo è un altro paragrafo normale allineato a sinistra e a destra. 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.

49.12.1 Spaziatura orizzontale automatica e separazione automatica dei paragrafi in righe

TeX utilizza una regola tipografica del passato, ormai condannata generalmente, secondo cui è necessario aumentare lo spazio che divide la fine di un periodo dall'inizio del successivo. Si pone rimedio a questa anomalia utilizzando l'istruzione '**`\frenchspacing`**'. Si osservi l'esempio seguente, il cui risultato di composizione viene ingrandito per permettere di scorgere meglio la differenza nella spaziatura orizzontale alla fine del primo periodo:

```
Prima frase. Seconda frase. \par
\frenchspacing
Prima frase. Seconda frase. \par
```

Prima frase. Seconda frase.

Prima frase. Seconda frase.

È bene sottolineare che '**`\frenchspacing`**' interviene a livello di paragrafi dal punto in cui viene utilizzato. Se successivamente si desidera ripristinare la spaziatura orizzontale «normale» per TeX, basta usare l'istruzione '**`\nonfrenchspacing`**'.

TeX impagina automaticamente i paragrafi, dividendo le righe dove ritiene più corretto, in base ai suoi calcoli, considerando le interruzioni di riga nel sorgente come equivalenti a uno spazio orizzontale, suddividendo eventualmente le parole in sillabe (come viene spiegato meglio in una sezione successiva). Tuttavia, ci sono spazi che fanno parte di un nome o che comunque si vuole evitare siano utilizzati per dividere in quel punto il testo. Per evitarlo si può inserire la tilde ('**`~`**') che rappresenta uno spazio non interrompibile. L'esempio seguente mostra il caso in cui si vuole evitare di separare il titolo da un nome:

```
Ing.~Bianchi
```

Quando non si usa la spaziatura orizzontale uniforme che si otterrebbe con '**`\frenchspacing`**', l'inserzione di questi spazi non interrompibili evita anche l'allargamento dello spazio quando, in situazioni come quella dell'esempio, sarebbe inopportuno. In alternativa, sempre allo scopo di ottenere uno spazio di ampiezza normale dopo un punto che non conclude una frase, si può usare il simbolo di controllo '**`\`**<*SP*>' (barra obliqua inversa, spazio).

49.12.2 Rientri particolari

Attraverso due variabili, a cui si accede per mezzo delle parole di controllo '**`\hangindent`**' e '**`\hangafter`**', è possibile organizzare un paragrafo in modo che una parte iniziale o finale delle righe che lo compongono sia rientrata rispetto al margine sinistro, oppure rispetto al margine destro.

Un valore positivo, esprimente una lunghezza, contenuto nella variabile controllata dalla parola di controllo '**`\hangindent`**', indica un rientro verso destra della stessa lunghezza rispetto al margine sinistro; al contrario, un valore negativo indica un rientro verso sinistra dal margine destro. Un valore positivo, esprimente una quantità di righe, contenuto nella variabile controllata dalla parola di controllo '**`\hangafter`**', indica dopo quante righe far intervenire il rientro dichiarato con '**`\hangafter`**'; invece, un valore negativo indica per quante righe iniziali deve valere il rientro. Si osservi l'esempio seguente:

```
\hangindent=5cm
\hangafter=2
Paragrafo in cui, dopo le prime due righe, si passa a un
rientro sinistro di cinque centimetri. 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 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.

\hangindent=-5cm
\hangafter=-2
Paragrafo in cui le prime due righe hanno un rientro destro
di cinque centimetri. 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 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.
```

La composizione che si ottiene potrebbe contenere degli errori estetici e degli errori nella separazione in sillabe delle parole. Questi problemi vengono trattati in seguito:

Paragrafo in cui, dopo le prime due righe, si passa a un rientro sinistro di
cinque centimetri. 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 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.

Paragrafo in cui le prime due righe hanno
un rientro destro di cinque centimetri. 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 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.

Dall'esempio non si vede, per cui è bene sottolineare che le parole di controllo '**`\hangindent`**' e '**`\hangafter`**' hanno effetto esclusivamente nel paragrafo in cui vengono utilizzate.

Un modo più complesso di definire un paragrafo è disponibile attraverso la parola di controllo '**`\parshape`**', da utilizzare secondo la sintassi seguente:

```
\parshape n_righe rientro_1 lunghezza_1 [rientro_2 lunghezza_2]...
```

In pratica, '**`\parshape`**' permette di descrivere il rientro sinistro e la lunghezza delle prime ***n*** righe; in base alla quantità di righe indicate, altrettante devono essere le coppie delle indicazioni sulla lunghezza del rientro e sulla lunghezza della riga relativa. Si osservi l'esempio seguente:

```
\parshape 3 2cm 5cm 4cm 6cm 6cm 7cm
Paragrafo in cui, la prima riga \'e rientrata di due
centimetri dal margine sinistro ed \'e lunga cinque
centimetri, la seconda riga \'e rientrata di quattro
centimetri ed \'e lunga sei centimetri, la terza riga \'e
rientrata di sei centimetri ed \'e lunga sette centimetri.
Le righe successive sono impaginate come la terza riga.
```

```
Paragrafo normale. 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 bla bla
```

Paragrafo in cui, la prima riga
è rientrata di due centimetri dal margine
sinistro ed è lunga cinque centimetri, la seconda riga è rientrata di quattro centimetri ed è lunga sei centimetri, la terza riga è rientrata di sei centimetri ed è lunga sette centimetri. Le righe successive sono impaginate come la terza riga.

Paragrafo normale. 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 bla bla bla bla bla bla

È importante osservare che la prima riga risulta avere un rientro maggiore, pari al valore restituito da '**\parindent**'. Si osservi lo stesso esempio, con l'eliminazione del rientro normale dei paragrafi:

```
\noindent
\parshape 3 2cm 5cm 4cm 6cm 6cm 7cm
Paragrafo in cui, la prima riga \'e rientrata di due
centimetri dal margine sinistro ed \'e lunga cinque
centimetri, la seconda riga \'e rientrata di quattro
centimetri ed \'e lunga sei centimetri, la terza riga \'e
rientrata di sei centimetri ed \'e lunga sette centimetri.
Le righe successive sono impaginate come la terza riga.

Paragrafo normale. 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 bla bla
```

Paragrafo in cui, la prima riga è
rientrata di due centimetri dal margine
sinistro ed è lunga cinque centimetri, la seconda riga è rientrata di quattro centimetri ed è lunga sei centimetri, la terza riga è rientrata di sei centimetri ed è lunga sette centimetri. Le righe successive sono impaginate come la terza riga.

Paragrafo normale. 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 bla bla bla bla bla bla

49.12.3 Paragrafi etichettati

La macro standard '**\item**' consente di porre all'inizio di un paragrafo un'etichetta, cosa che si fa normalmente per realizzare degli elenchi:

```
\item{etichetta}
```

Il paragrafo che si ottiene ha una struttura particolare: L'etichetta viene inserita in modo che termini subito prima della posizione del rientro della prima riga, ovvero subito prima di '**\hoffset**'+'**\leftskip**'+'**\parindent**'; il testo successivo inizia dal rientro e nelle righe successive mantiene quel rientro iniziale. La parola di controllo '**\noindent**' non ha effetto.

Esiste anche un'altra parola di controllo simile, che ha lo scopo di generare un paragrafo dello stesso genere, rientrato del doppio del valore restituito da '**\parindent**'. Si tratta di '**\itemitem**', il cui utilizzo è quindi conveniente solo quando si usa normalmente il rientro della prima riga, altrimenti l'effetto generato è lo stesso di '**\item**'. Si osservino i due esempi seguenti che differiscono per la presenza o meno del rientro della prima riga.

```
Questo \'e un paragrafo normale che introduce un elenco, 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:
\item{1)} questa \'e la prima voce dell'elenco;
\item{2)} questa \'e la seconda voce dell'elenco, che
contiene molte informazioni, bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla;
\item{3)} questa \'e la terza voce dell'elenco che si
articola in voci di livello inferiore:
\itemitem{a)} questa \'e la prima sotto-voce del terzo
punto;
\itemitem{b)} questa \'e la seconda sotto-voce del terzo
```



```
punto;
\item{4)} questo \'e la quarta e ultima voce dell'elenco
principale.
```

Questo è un paragrafo normale che introduce un elenco, 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:

1) questa è la prima voce dell'elenco;
2) questa è la seconda voce dell'elenco, che contiene molte informazioni, bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla;
3) questa è la terza voce dell'elenco che si articola in voci di livello inferiore:
   a) questa è la prima sotto-voce del terzo punto;
   b) questa è la seconda sotto-voce del terzo punto;
4) questo è la quarta e ultima voce dell'elenco principale.

Come si può vedere, le macroistruzioni '**\item**' e '**\itemitem**' introducono un paragrafo nuovo, indipendentemente dal fatto che siano o meno separate dal testo precedente. Nell'esempio seguente si mostra cosa accade se si elimina il rientro della prima riga dei paragrafi:

```
\parindent=0pt
Questo \'e un paragrafo normale che introduce un elenco, 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:
\item{1)} questa \'e la prima voce dell'elenco;
\item{2)} questa \'e la seconda voce dell'elenco, che
contiene molte informazioni, bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla;
\item{3)} questa \'e la terza voce dell'elenco che si
articola in voci di livello inferiore:
\itemitem{a)} questa \'e la prima sotto-voce del terzo
punto;
\itemitem{b)} questa \'e la seconda sotto-voce del terzo
punto;
\item{4)} questo \'e la quarta e ultima voce dell'elenco
principale.
```

Questo è un paragrafo normale che introduce un elenco, 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:
1) questa è la prima voce dell'elenco;
2) questa è la seconda voce dell'elenco, che contiene molte informazioni, bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla;
3) questa è la terza voce dell'elenco che si articola in voci di livello inferiore:
a) questa è la prima sotto-voce del terzo punto;
b) questa è la seconda sotto-voce del terzo punto;
4) questo è la quarta e ultima voce dell'elenco principale.

Volendo approfondire un po' di più l'uso di queste macroistruzioni, si può osservare che '**\item**' assegna a '**\hangindent**' il valore di '**\parindent**'; inoltre assegna a '**\hangafter**' il valore uno. Al termine, tutto viene ripristinato, in modo da non coinvolgere il paragrafo successivo. Nello stesso modo interviene '**\itemitem**', assegnando però a '**\hangindent**' il doppio di '**\parindent**'.

49.12.4 Attuazione dei comandi riferiti ai paragrafi

Le istruzioni che definiscono l'aspetto del paragrafo, vengono prese in considerazione quando il paragrafo è stato analizzato completamente; pertanto, se in un paragrafo si modifica più volte un certo valore che ne controlla la forma, solo l'ultimo valore disponibile è quello che viene preso in considerazione. Si osservi l'esempio seguente:

```
\baselineskip=5cm
Questo paragrafo viene composto con una distanza tra la base
delle righe di 14 punti. Provare per credere. 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 bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla.
\baselineskip=14pt
```

Questo paragrafo viene composto con una distanza tra la base delle righe di 14 punti. Provare per credere. 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 bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla.

In pratica, all'inizio viene dichiarata una distanza di 5 cm, mentre alla fine, prima che sia concluso definitivamente il paragrafo, si di-

chiara nuovamente una distanza di soli 14 punti. L'ultima istruzione che modifica il valore, rimanendo nell'ambito del paragrafo, è quella che viene presa in considerazione.

Il problema può sembrare di poco conto; tuttavia, quando si usano i raggruppamenti per evitare di estendere l'effetto di un comando particolare anche ai paragrafi successivi, le cose si complicano un po'. Si osservi l'esempio seguente:

```
{\baselineskip=5cm Questo paragrafo che dovrebbe avere una
distanza tra la base delle righe di ben cinque centimetri,
ma qualcosa non funziona come si vorrebbe. 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.}
```

Questo paragrafo che dovrebbe avere una distanza tra la base delle righe di ben cinque centimetri, ma qualcosa non funziona come si vorrebbe. 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.

Il raggruppamento limita l'effetto della modifica dello spazio tra le righe, controllato da '**`\baselineskip`**', all'interno del paragrafo. Tuttavia, il paragrafo finisce dopo la parentesi graffa di chiusura, ovvero quando viene ripristinato il valore precedente per la distanza tra le righe. Per risolvere il problema occorre concludere il paragrafo **prima** della parentesi graffa finale:

```
{\baselineskip=0.5cm Questo paragrafo ha effettivamente una
distanza tra la base delle righe di mezzo centimetro,
perch\'e prima della parentesi graffa finale si conclude
esplicitamente il paragrafo.\par}
```

Questo paragrafo ha effettivamente una distanza tra la base delle righe di mezzo centimetro, perché prima della parentesi graffa finale si conclude esplicitamente il paragrafo.

49.12.5 Tolleranza estetica

«

Quando la composizione che si genera non può rispettare gli spazi e le elasticità fissati, TeX calcola un indicatore, definibile come penalità estetica, per rappresentare la «bruttezza» del risultato che si ottiene. Per la precisione, si distinguono due casi: la necessità di introdurre una quantità di spazio eccessiva, tra le parole o tra le righe, oppure la necessità di andare leggermente oltre il margine.

Quando si superano le tolleranze orizzontali stabilite, si ottiene nella composizione finale un segno evidente sul margine destro, corrispondente a un rettangolino nero.

La parola di controllo '**`\hbadness`**' permette di definire il livello consentito per l'espansione delle righe. Il valore predefinito è 1000 e rappresenta un livello qualitativo molto buono. Elevando questo valore, si ottengono meno segnalazioni; si può arrivare fino a un massimo di 10000, con cui si esclude qualunque segnalazione di cattiva qualità estetica dovuta a un allargamento eccessivo delle righe.

Una riga può eccedere il limite del margine destro per una lunghezza massima pari al valore che si ottiene dalla parola di controllo '**`\hfuzz`**'. Se anche questo limite viene superato, si crea un altro tipo di problema estetico, controllato dalla parola di controllo '**`\tolerance`**'. Anche in questo caso, se si assegna alla tolleranza in eccesso un valore pari a 10000, si annullano tutte le segnalazioni di errore.

La qualità verticale è controllata dal valore contenuto in '**`\vbadness`**' e nello stesso modo esiste la possibilità di eccedere l'altezza del testo del valore di '**`\vfuzz`**'.

```
hbadness = \the\hbadness \par
hfuzz = \the\hfuzz \par
vbadness = \the\vbadness \par
vfuzz = \the\vfuzz \par
tolerance = \the\tolerance \par
```

hbadness = 1000
hfuzz = 0.1pt
tolerance = 200
vbadness = 1000
vfuzz = 0.1pt

L'esempio precedente consente di verificare i valori predefiniti abbinati alla tolleranza estetica del testo, nella sua distribuzione orizzontale e verticale. Le istruzioni che seguono, consentono di annullare tutte le segnalazioni di carenze estetiche, consentendo però di generare un risultato pessimo:

```
\hbadness=10000
\vbadness=10000
\tolerance=10000
```

In alternativa, si possono lasciare invariati i valori di tolleranza estetica, eliminando semplicemente l'indicazione del problema nella composizione finale. L'inserimento del rettangolo nero è controllato dalla parola di controllo '**`\overfullrule`**', a cui si assegna la grandezza della linea con cui rappresentare questo rettangolo. Indicando un valore nullo, si elimina tutto questo:

```
\overfullrule=0pt
```

È utile sapere interpretare i messaggi di avvertimento che vengono generati da TeX durante la composizione di testo con carenze estetiche. La tabella 49.161 riepiloga le varie situazioni.

Tabella 49.161. Messaggi di errore riferiti a carenze estetiche.

Tipo di messaggio	Significato
'**`Underfull \hbox`**'	La spaziatura orizzontale è eccessiva.
'**`Overfull \hbox`**'	Lo spazio orizzontale a disposizione è stato superato.
'**`Underfull \vbox`**'	La spaziatura verticale è eccessiva.
'**`Overfull \vbox`**'	Lo spazio verticale a disposizione è stato superato.
'**`badness`** ***n***'	Livello di carenza estetica: 10000 è il massimo a essere annotato.

49.12.6 Sillabazione

«

In condizioni normali, TeX gestisce la separazione in sillabe per la lingua inglese. Eventualmente, è possibile indicare delle eccezioni attraverso l'uso della parola di controllo '**`\hyphenation`**':

`\hyphenation{`***parola_suddivisa***...`}`

Per esempio, per annotare la suddivisione di «cartellone» e «manuale», si può indicare l'istruzione seguente:

```
\hyphenation{car-tel-lo-ne ma-nu-a-le}
```

È possibile anche usare una separazione facoltativa attraverso il simbolo di controllo '**`\-`**' all'interno di una parola in una posizione precisa. Tuttavia, l'uso di queste istruzioni, annulla la possibilità di formarsi un legato. Per esempio, scrivendo '**`dif\-fi\-ci\-le`**' si impedisce la formazione del legato tra la coppia di lettere «f».

La frequenza della separazione delle parole alla fine delle righe può essere regolata tramite il valore attribuito a una variabile, cui si accede attraverso la parola di controllo '**`\pretolerance`**': Valori bassi producono una separazione frequente, mentre valori elevati riducono l'uso della suddivisione delle parole. In particolare, se si assegna il valore 10000, non avviene alcuna separazione in sillabe.

`\pretolerance=`***n***

Il valore predefinito di '**`\pretolerance`**' è 100.

È importante sottolineare che '**`\pretolerance`**' interviene a livello di paragrafo, per cui non è possibile isolare un gruppo all'interno di un paragrafo, perché queste modifiche non avrebbero effetto.

Nella sezione 49.1 è descritto brevemente come abilitare la separazione delle parole in sillabe per una lingua differente. Supponendo che la sillabazione italiana sia abbinata al numero 1, è sufficiente iniziare il proprio documento con l'istruzione seguente:

```
\language 1
```

49.12.7 Spaziature verticali

È già stato descritto il significato delle parole di controllo '**\baselineskip**' e '**\parskip**'. Entrambe le variabili a cui fanno riferimento queste parole di controllo possono contenere una lunghezza elastica, anche se normalmente '**\baselineskip**' non prevede tolleranze.

Oltre allo spazio verticale tra i paragrafi, controllato dal valore restituito dalla parola di controllo '**\parskip**', si possono inserire degli spazi aggiuntivi[8] attraverso due parole di controllo alternative, '**\vskip**' e '**\vglue**':

```
\vskip lunghezza_elastica
```

```
\vglue lunghezza_elastica
```

Il secondo tra i due tipi di inserimento, corrispondente alla parola di controllo '**\vglue**', è il più semplice, perché inserisce lo spazio verticale in modo incondizionato: se non c'è lo spazio sufficiente nella pagina in cui appare, viene messo integralmente all'inizio della pagina successiva. Invece, nel caso di '**\vskip**', lo spazio viene inserito effettivamente solo se c'è lo spazio sufficiente nella pagina in cui appare, provocando un salto pagina se questo spazio manca, ma in tal caso non viene più inserito all'inizio della pagina successiva.

È importante osservare che in entrambe le situazioni, se lo spazio alla fine della pagina non basta, si provoca un salto pagina. Se è attiva la modalità di allineamento verticale del testo, corrispondente alla parola di controllo '**\normalbottom**', ciò significa che il testo precedente viene allungato per completare la pagina, così come avviene quando si inserisce la parola di controllo '**\eject**'.

A titolo informativo, vale la pena di vedere in che modo si possa ottenere l'inserimento di uno spazio incondizionato, come fa '**\vglue**', ma utilizzando solo '**\vskip**':

```
\null\par
\vskip …
```

In pratica, l'istruzione '**\null**' serve a iniziare un paragrafo (passando così in modalità orizzontale). Il paragrafo deve essere terminato, quindi si può inserire '**\vskip**' con l'indicazione dello spazio da saltare. L'istruzione '**\null**' corrisponde a un carattere nullo, come un gruppo vuoto, che però permette di iniziare un paragrafo (viene descritta in seguito).

È già stato fatto osservare che il valore predefinito che si ottiene dall'espansione di '**\parskip**' è '**0pt plus 1pt**'. Volendo mantenere la coerenza con la spaziatura tra i paragrafi, quando si usano le parole di controllo '**\vskip**' e '**\vglue**', conviene mantenere la stessa elasticità in espansione, con l'aggiunta eventuale di un'elasticità equivalente in contrazione:

```
\vskip lunghezza plus 1pt minus 1pt
```

```
\vglue lunghezza plus 1pt minus 1pt
```

In effetti, TeX offre già delle parole di controllo che si traducono in salti standardizzati. Si tratta di '**\smallskipamount**', '**\medskipamount**' e '**\bigskipamount**'. Si può sperimentare facilmente a cosa corrispondono:

```
smallskipamount = \the\smallskipamount \par
medskipamount = \the\medskipamount \par
bigskipamount = \the\bigskipamount \par
```

smallskipamount = 3.0pt plus 1.0pt minus 1.0pt
medskipamount = 6.0pt plus 2.0pt minus 2.0pt
bigskipamount = 12.0pt plus 4.0pt minus 4.0pt

Queste parole di controllo sono fatte per essere sostituite agli argomenti delle istruzioni che provocano un avanzamento verticale. Per esempio,


```
\vskip\smallskipamount
```

provoca quello che viene considerato convenzionalmente un piccolo avanzamento verticale.

Per semplificare ulteriormente la gestione dei salti verticali, con la parola di controllo '**\vskip**', sono disponibili delle macro che richiamano direttamente l'entità dello spazio da inserire. Si tratta di '**\smallskip**', '**\medskip**' e '**\bigskip**', corrispondenti in pratica a '**\vskip\smallskipamount**', '**\vskip\medskipamount**' e '**\vskip\bigskipamount**'.

Quando si vuole indicare uno salto verticale di lunghezza indefinita, si utilizzano le parole di controllo '**\vfil**' e '**\vfill**'. Entrambe generano un salto dal punto in cui si trovano, fino alla fine della pagina, qualsiasi sia la distanza, ma se la parola di controllo appare più volte, lo spazio necessario viene distribuito equamente. Si osservi l'esempio seguente in cui si simula una pagina di piccole dimensioni:

```
\hsize=4cm
\vsize=7cm
Questa \'e una pagina fittizia in cui il testo ha a
disposizione solo quattro per sette centimetri. \par
\vfil
I paragrafi sono spaziati verticalmente in modo da riempire
la pagina. \par
\vfil
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. \par
\vfil
Fine della pagina. \par
\eject
```

Questa è una pagina fittizia in cui il testo ha a disposizione solo quattro per sette centimetri.

I paragrafi sono spaziati verticalmente in modo da riempire la pagina.

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.

Fine della pagina.

La stessa cosa si potrebbe ottenere sostituendo le parole di controllo '**\vfil**' con '**\vfill**':

```
\hsize=4cm
\vsize=7cm
Questa \'e una pagina fittizia in cui il testo ha a
disposizione solo quattro per sette centimetri. \par
\vfill
I paragrafi sono spaziati verticalmente in modo da riempire
la pagina. \par
\vfill
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. \par
\vfill
Fine della pagina. \par
\eject
```

Tuttavia, '**\vfill**' è più elastico rispetto a '**\vfil**'. Si osservi cosa accade se si inserisce un solo '**\vfill**' tra altri '**\vfil**':

```
\hsize=4cm
\vsize=7cm
Questa \'e una pagina fittizia in cui il testo ha a
disposizione solo quattro per sette centimetri. \par
\vfil
I paragrafi sono spaziati verticalmente in modo da riempire
la pagina. \par
\vfill
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. \par
\vfil
Fine della pagina. \par
\eject
```

Questa è una pagina fittizia in cui il testo ha a disposizione solo quattro per sette centimetri.
I paragrafi sono spaziati verticalmente in modo da riempire la pagina.

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.
Fine della pagina.

Esiste un altro tipo di spazio verticale elastico, controllato dalla parola di controllo '**`\vss`**'. Si ottiene un risultato molto simile a '**`\vfil`**', con la differenza che in questo caso viene intesa esplicitamente anche un'elasticità in contrazione di livello equivalente.

Le parole di controllo '**`\vfil[l]`**' e '**`\vss`**' corrispondono in pratica alle definizioni seguenti:

```
\vfil | \vskip 0pt plus 1fil
\vfill | \vskip 0pt plus 1fill
\vss | \vskip 0pt plus 1fil minus 1fil
```

49.13 Righe

«

Il testo di una riga viene generato posando le lettere su una linea invisibile che corrisponde alla base della stessa. Nella sezione 47.2 è affrontato in generale l'argomento relativo alle dimensioni di un carattere. Per quanto riguarda TeX le cose si semplificano come si vede nella figura 49.173.

Figura 49.173. Le dimensioni del carattere secondo TeX.

Le righe di un paragrafo sono distanziate, da una basa alla successiva, della distanza fissata con '**`\baselineskip`**'. Ma questo non può impedire che si formino degli accavallamenti, soprattutto se si ingrandisce il corpo del carattere. Pertanto, entra in gioco il valore contenuto nella variabile '**`\lineskiplimit`**' che rappresenta l'interlinea minima, ovvero lo spazio minimo tra la profondità massima della riga precedente e l'altezza massima della riga successiva. Se si supera questo valore, viene inserita un'interlinea pari alla lunghezza contenuta nella variabile '**`\lineskip`**'. Si osservi l'esempio seguente in cui a un certo punto viene cambiato il corpo del carattere, per dimostrare l'adattamento che si ottiene nella distanza tra le righe:

```
\font\prova=cmss17 scaled 2000

Questo \'e un paragrafo di prova in cui, a un certo punto,
si usa un carattere pi\'u grande del solito, provocando un
aumento automatico della distanza tra le righe, {\prova per
evi\-ta\-re accavallamenti}. 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. \par
```

Questo è un paragrafo di prova in cui, a un certo punto, si usa un carattere più grande del solito, provocando un aumento automatico della distanza tra le righe, per evitare accavallamenti. 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.

È possibile sollevare o abbassare una parte del testo, attraverso le istruzioni '**`\raise`**' e '**`\lower`**'. Queste devono essere seguite da una «scatola», contenente ciò che si vuole sollevare o abbassare rispetto alla base della riga in cui ci si trova. Il concetto di scatola viene spiegato nella sezione 49.14.

`\raise` ***altezza scatola***

`\lower` ***altezza scatola***

L'esempio seguente mostra due situazioni, in cui si sposta una parte del testo in alto e in basso rispetto alla base normale delle righe:

```
Nel testo c'\'e una parte leggermente sollevata di
\raise 5pt \hbox{cinque punti} e un'altra parte
leggermente abbassata di \lower 3pt \hbox{tre punti}
tipografici.
```

Nel testo c'è una parte leggermente sollevata di cinque punti e un'altra parte leggermente abbassata di tre punti tipografici.

Quando una riga singola ha una sua propria autonomia nel testo che si vuole scrivere, si possono utilizzare una serie di parole di controllo per gestirne la collocazione e l'aspetto.

La prima cosa da analizzare è la possibilità di interrompere il flusso normale di un paragrafo, imponendo di iniziare una riga nuova. Ciò si ottiene con la parola di controllo '**`\break`**':

```
Questo \'e un paragrafo normale, contenente alcune
informazioni, bla bla bla bla bla bla bla bla bla bla
bla,\break ma questa riga deve apparire sempre da sola, pur
senza dover iniziare un paragrafo nuovo.
```

Questo è un paragrafo normale, contenente alcune informazioni, bla bla bla bla bla bla bla bla bla bla bla bla,
ma questa riga deve apparire sempre da sola, pur senza dover iniziare un paragrafo nuovo.

L'esempio mostra proprio una situazione di questo tipo, dove si può osservare che il testo della riga incompleta viene allungato per raggiungere il margine destro. Per evitare che succeda questo, si utilizza uno spazio orizzontale allungabile, che può essere ottenuto a scelta da tre parole di controllo: '**`\hfil`**', '**`\hfill`**' e '**`\hfilll`**'. Così come avviene negli spazi verticali, la prima di queste parole di controllo ha un'elasticità minore rispetto alle altre, così come la seconda ha un'elasticità maggiore della prima e minore della terza. Generalmente, si usa '**`\hfill`**' prima di '**`\break`**', come si vede dall'esempio seguente:

```
Questo \'e un paragrafo normale, contenente alcune
informazioni, bla bla bla bla bla bla bla bla bla bla
bla,\hfill\break ma questa riga deve apparire sempre da
sola, pur senza dover iniziare un paragrafo nuovo.
```

Questo è un paragrafo normale, contenente alcune informazioni, bla bla bla bla bla bla bla bla bla bla bla,
ma questa riga deve apparire sempre da sola, pur senza dover iniziare un paragrafo nuovo.

Quando si vuole controllare una riga completamente autonoma, si possono usare le macroistruzioni '**`\leftline`**', '**`\rightline`**', '**`\centerline`**' e '**`\line`**'. La prima allinea a sinistra il testo, la seconda lo allinea a destra, la terza lo allinea al centro, l'ultima lo stira in modo da allinearlo simultaneamente a sinistra e a destra. Si osservi l'esempio:

```
\leftline{Questa riga \'e allineata a sinistra.}
\centerline{Questa riga \'e allineata al centro.}
\rightline{Questa riga \'e allineata a destra.}
\line{Questa riga \'e allineata a sinistra e a destra.}
```

Questa riga è allineata a sinistra.
Questa riga è allineata al centro.
Questa riga è allineata a destra.
Questa riga è allineata a sinistra e a destra.

Si potrebbe essere portati a pensare che la macroistruzione '**`\line`**' serva a poco, perché l'effetto che si ottiene è abbastanza sgradevole. In realtà, la possibilità di inserire degli spazi orizzontali allungabili consente di creare degli effetti molto utili. Si osservi l'esempio seguente in cui si fa uso di '**`\hfil`**' per separare degli allineamenti indipendenti:

```
\line{Sinistra \hfil Centro \hfil Destra}
```

Sinistra Centro Destra

In particolare, '**\headline**' e '**\footline**' funzionano nello stesso modo, perché sono controllate in pratica dalla parola di controllo '**\line**'.

Si osservi che la definizione delle macroistruzioni '**\[left|center|right]line**' isola il testo in un raggruppamento, come se fosse racchiuso tra parentesi graffe.

49.13.1 Spazi orizzontali

È possibile inserire degli spazi orizzontali ben definiti attraverso la parola di controllo '**\hskip**', a cui si assegna l'indicazione di una lunghezza elastica:

```
\hskip lunghezza_elastica
```

Come nel caso di '**\vskip**', l'indicazione di un'elasticità in estensione, consente in pratica di allargare lo spazio in modo indefinito.

L'esempio seguente mostra diversi casi di utilizzo di '**\hskip**', in una situazione in cui la riga che lo contiene viene allineata simultaneamente a sinistra e a destra:

```
\line{Riga allineata a \hskip 2cm sinistra e a destra.}
\line{Riga allineata a \hskip 2cm plus 1pt sinistra e a destra.}
\line{Riga allineata \hskip 2cm plus 1pt a sinistra \hfil e a destra.}
\line{Riga allineata \hskip 0pt plus 1fil a sinistra \hfil e a destra.}
\line{Riga allineata \hskip 0pt plus 2fil a sinistra \hfil e a destra.}
```

Riga allineata a sinistra e a destra.
Riga allineata a sinistra e a destra.
Riga allineata a sinistra e a destra.
Riga allineata a sinistra e a destra.
Riga allineata a sinistra e a destra.

Esiste un altro tipo di spazio orizzontale molto elastico, controllato dalla parola di controllo '**\hss**'. Si ottiene un risultato molto simile a '**\hfil**', con la differenza che in questo caso viene intesa esplicitamente anche un'elasticità in contrazione di livello equivalente.

Le parole di controllo '**\hfil[l[l]]**' e '**\hss**' corrispondono in pratica alle definizioni seguenti:

```
\hfil  | \hskip 0pt plus 1fil
\hfill | \hskip 0pt plus 1fill
\hfilll | \hskip 0pt plus 1filll
\hss | \hskip 0pt plus 1fil minus 1fil
```

Esiste infine un gruppo di parole di controllo che consentono di inserire spazi orizzontali rigidi o poco elastici, corrispondenti all'elenco seguente:

- `\enspace`
  rappresenta uno spazio orizzontale rigido di mezzo quadratone;
- `\enskip`
  rappresenta uno spazio orizzontale leggermente elastico di mezzo quadratone;
- `\quad`
  rappresenta uno spazio orizzontale leggermente elastico di un quadratone;
- `\qquad`
  rappresenta uno spazio orizzontale leggermente elastico di due quadratoni;

- `\thinspace`
  in modalità orizzontale, introduce uno spazio rigido molto sottile, pari a un sesto di quadratone;
- `\negthinspace`
  in modalità orizzontale, introduce uno spazio negativo rigido molto sottile, pari a un sesto di quadratone.

Se si intende iniziare un paragrafo con un rientro prodotto da un'istruzione '**\hskip**', o altre che comunque producono uno spazio orizzontale determinato, occorre tenere presente la presenza del rientro definito da '**\parindent**'. In pratica, se si vuole iniziare un paragrafo con un rientro iniziale di due centimetri usando l'istruzione '**\hskip 2cm**', occorre fare in questo modo:

```
\noindent \hskip 2cm Bla bla bla...
```

Diversamente, il rientro finale risulterebbe aumentato del valore di '**\parindent**'.

Un'ultima cosa da considerare sono gli spazi orizzontali della stessa ampiezza di un certo carattere. Quando per qualche ragione si ha la necessità di incolonnare delle informazioni, soprattutto dati numerici, può essere utile la macroistruzione '**\phantom**', la quale genera effettivamente lo spazio orizzontale equivalente al carattere indicato:

```
\phantom{x}
```

Anche se l'utilizzo più importante riguarda i numeri, viene proposto qualcosa di diverso, per consentire di verificare visivamente che lo spazio viene calcolato in modo corretto:

```
\font\prova=cmss12
\prova
Ciao ciao ciao ciao \par
\hskip 0pt\phantom{C}iao ciao c\phantom{i}ao ciao \par
```

Ciao ciao ciao ciao
iao ciao c ao ciao

Come si vede, si utilizza un carattere senza grazie, in modo da poter verificare che l'inserzione dello spazio corrispondente a una lettera «i» viene fatta correttamente, così come è corretto lo spazio corrispondente alla lettera «C» maiuscola. L'istruzione iniziale '**\hskip 0pt**' serve a imporre che si tenga conto dello spazio iniziale, perché altrimenti verrebbe perduto nell'inizio del paragrafo.

49.13.2 Spazio rigido

Lo spazio rigido per definizione viene richiesto attraverso l'istruzione '**\kern**' che richiede semplicemente l'indicazione della distanza da compiere:

```
\kern lunghezza
```

Lo spazio in questione può essere orizzontale o verticale, in base al contesto. Pertanto, se ci si trova in modalità orizzontale, si ottiene uno spazio orizzontale; diversamente si ottiene uno spazio verticale. Per questo, quando si vuole scrivere una macroistruzione con cui iniziare sicuramente la modalità orizzontale e inserire quindi uno spazio rigido, si usa spesso '**\leavevmode**' per questo scopo.

Lo spazio rigido viene usato a volte in modo negativo, per avvicinare delle lettere o altri simboli, garantendo che non avvenga la separazione della riga in quel punto. Infatti, gli spazi rigidi che si ottengono con '**\kern**' non sono separabili. L'esempio tipico di utilizzo di '**\kern**' per avvicinare le lettere è quello della macroistruzione '**\TeX**', dichiarata nel modo seguente:

```
\def\TeX{T\kern -.1667em\lower .5ex\hbox {E}\kern -.125emX}
```

Come si vede, non c'è alcun bisogno di richiedere espressamente il passaggio alla modalità orizzontale, perché le istruzioni '**\kern**' appaiono dentro un testo che ha già fatto questo in ogni caso.

49.13.3 Linee guida

Per realizzare delle linee guida orizzontali, allo scopo di accompagnare la lettura verso qualcosa, si possono usare delle parole di controllo che contengono implicitamente un'elasticità in estensione molto elevata:

```
\hrulefill
```

```
\dotfill
```

```
\leftarrowfill | \rightarrowfill
```

Nel primo caso si ottiene una linea continua che unisce due parti di un testo separato; nel secondo si ottiene invece una guida fatta di puntini; nel terzo caso si genera una linea continua con una piccola freccia all'estremità sinistra o all'estremità destra.

Tuttavia, oltre a questi casi di guide abbastanza comuni, è possibile definire le proprie con una tra le istruzioni seguenti:

```
\leaders\hbox [to distanza] {espressione_guida} \hfill
```

```
\cleaders\hbox [to distanza] {espressione_guida} \hfill
```

```
\xleaders\hbox [to distanza] {espressione_guida} \hfill
```

La differenza tra le parole di controllo '**\leaders**', '**\cleaders**' e '**\xleaders**', sta nel fatto che la seconda dovrebbe centrare la linea guida, mentre l'ultima dovrebbe estenderla in modo da usare tutto lo spazio a disposizione. In pratica è difficile comprendere bene la differenza da un punto di vista visivo, per cui si utilizza in genere solo il primo tipo, salvo provare gli altri per verificare l'effetto finale che si può ottenere in alternativa.

Generalmente, nell'espressione che si utilizza per creare la guida si inseriscono parole di controllo per elasticizzare la distanza dei simboli utilizzati.

È importante sottolineare che TeX, dopo la parola di controllo '**\[c|x]leader**' vuole necessariamente una scatola '**\hbox**' e alla fine vuole uno spazio elastico di tipo '**\hfil[l[l]]**'. Nel modello sintattico non è stata mostrata la possibilità di usare diversi spazi orizzontali elastici, perché in pratica ci si limita sempre solo a usare '**\hfill**'.

Nel caso si utilizzi '**\hbox**' con l'indicazione della larghezza, l'effetto che si ottiene è quello di distanziare precisamente gli elementi simbolici che compongono la linea guida. In questo senso, l'elasticità che gli si attribuisce all'interno del raggruppamento, serve a collocarli correttamente nella scatola.

L'esempio seguente mostra diversi casi di utilizzo di queste linee guida messi a confronto. In particolare, si creano delle linee guida speciali, in cui si usa l'asterisco come simbolo ripetuto:

```
linea semplice: \hrulefill fine della linea \par
puntini: \dotfill fine dei puntini \par
linea con freccia a destra: \rightarrowfill fine della linea \par
linea con freccia a sinistra: \leftarrowfill fine della linea \par
asterischi normali: \leaders\hbox {*} \hfill fine degli asterischi \par
asterischi distanziati: \leaders\hbox to 1cm {\hss * \hss} \hfill
 fine degli asterischi \par
asterischi normali: \cleaders\hbox {*} \hfill fine degli asterischi \par
asterischi distanziati: \cleaders\hbox to 1cm {\hss * \hss} \hfill
 fine degli asterischi \par
asterischi normali: \xleaders\hbox {*} \hfill fine degli asterischi \par
asterischi distanziati: \xleaders\hbox to 1cm {\hss * \hss} \hfill
 fine degli asterischi \par
```


linea semplice: _____ fine della linea
puntini: ..... fine dei puntini
linea con freccia a destra: ——→ fine della linea
linea con freccia a sinistra: ←—— fine della linea
asterischi normali: ********** fine degli asterischi
asterischi distanziati: * * * * fine degli asterischi
asterischi normali: ********** fine degli asterischi
asterischi distanziati: * * * * fine degli asterischi
asterischi normali: ********** fine degli asterischi
asterischi distanziati: * * * * fine degli asterischi

Volendo creare una macroistruzione per una linea guida particolare, si può procedere come nell'esempio seguente, in cui si riprende uno dei casi già presentati sopra:

```
\def\asterischi{\leaders\hbox to 1cm {\hss * \hss} \hfill}
```

49.13.4 Linee

TeX consente di tracciare delle linee orizzontali o verticali, attraverso le parole di controllo '**\hrule**' e '**\vrule**'. La sintassi per queste istruzioni può essere abbastanza articolata:

```
\hrule [width larghezza] [height altezza] [depth profondità]
```

```
\vrule [width larghezza] [height altezza] [depth profondità]
```

La differenza più importante tra le due parole di controllo sta nel fatto che '**\hrule**' introduce implicitamente una separazione di paragrafi, mentre '**\vrule**' no.

Quando '**\hrule**' viene usato senza argomenti, si ottiene una linea orizzontale, appoggiata sulla base della riga, spessa 0,4 punti e larga quanto lo spazio a disposizione per il testo, ovvero quanto definito da '**\hsize**'; nello stesso modo, '**\vrule**' senza argomenti genera una linea verticale, larga 0,4 punti, che parte dalla base della riga e si innalza per l'altezza massima del contesto in cui si trova. Si osservi l'esempio:

```
Prima di una linea orizzontale; \hrule{} dopo una linea
orizzontale. \par
Prima di una linea verticale; \vrule{} dopo una linea
verticale. \par
```

Prima di una linea orizzontale;
dopo una linea orizzontale.
Prima di una linea verticale; | dopo una linea verticale.

Quando si usano gli argomenti con cui si possono controllare le caratteristiche di queste linee, è meglio pensare a dei rettangoli, dove l'unica differenza che conta è il fatto che '**\hrule**' conclude un paragrafo. Seguono altri esempi a questo proposito:

```
Linea orizzontale lunga 10 cm e alta 1 punto: \hrule width
10cm height 1pt \par
Linea verticale alta 1 cm e larga 2 punti: \vrule height 1cm
width 2pt \par
Rettangolino alto 5 mm e largo 3 mm: \vrule height 5mm width
3mm \par
Linea orizzontale nel testo: \vrule height 0pt depth 1pt
width 10cm \par
```

Linea orizzontale lunga 10 cm e alta 1 punto:
Linea verticale alta 1 cm e larga 2 punti:
Rettangolino alto 5 mm e largo 3 mm:
Linea orizzontale nel testo:

49.14 Scatole

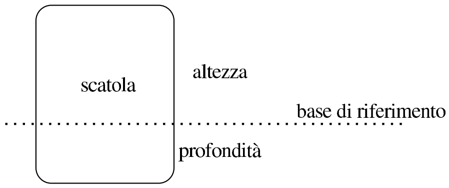
TeX tratta tutti gli oggetti da collocare nella composizione come tanti rettangoli, o scatole, di dimensione appropriata, distanziati in base a delle regole determinate. In questo senso, gli spazi che si inseriscono nel file sorgente, corrispondenti al carattere <*SP*>, comprese le tabulazioni orizzontali (il carattere <*HT*>), assieme alle righe vuote e a quelle bianche, sono solo la separazione logica delle istruzioni (intendendo anche il testo come istruzioni), pertanto non conta la loro quantità, essendo sufficiente che siano presenti dove serve.

Attraverso alcune parole di controllo è possibile creare della scatole di tipo orizzontale o verticale, per ottenere dei comportamenti par-

ticolari che vanno al di fuori della gestione normale dei paragrafi e delle righe.

Una scatola ha tre misure che ne definiscono la forma: una larghezza, un'altezza che parte dalla base della riga di riferimento e una profondità, che è rappresentata dall'altezza della scatola che si espande sotto la base di riferimento.

Figura 49.199. Dimensioni di una scatola.



Una scatola orizzontale è un rettangolo in cui si inserisce una sola riga di testo, dove l'altezza e la profondità dipendono dalle dimensioni del testo stesso. Una scatola verticale è invece un'area in cui il testo può scomporsi in più righe; inoltre, se la scatola viene inserita in una riga di testo, questa conclude il paragrafo.

Si utilizzano anche delle scatole vuote. Una scatola orizzontale vuota può servire per inserire un testo sporgente, a sinistra o a destra rispetto alla stessa; inoltre, può servire per imporre una certa altezza e profondità al testo della riga in cui viene inserita.

Una scatola orizzontale si definisce generalmente con la parola di controllo '**\hbox**' che può essere usata in due modi differenti:

\hbox {*espressione_stringa*}

\hbox to *larghezza* {*espressione_stringa*}

Nel primo caso si definisce una scatola orizzontale, contenente il testo che si ottiene dall'espansione dell'espressione racchiusa nel raggruppamento, larga quanto la giustezza del testo stesso; nel secondo caso, si impone la larghezza indicata.

È importante osservare che '**\hbox**' non è propriamente una macro e il suo argomento è precisamente un gruppo, per cui l'espressione contenuta è isolata dal testo successivo. Lo stesso dicasi per le parole di controllo che definiscono delle scatole verticale, descritte più avanti.

Esistono delle parole di controllo per la gestione delle scatole orizzontali vuote a vario titolo:

\llap {*espressione*}

\rlap {*espressione*}

\null

\strut

\mathstrut

Le parole di controllo '**\llap**' e '**\rlap**' definiscono un gruppo vuoto, in cui l'espressione del gruppo che viene indicato risulta sporgere rispettivamente a sinistra e a destra. I nomi usati per queste parole di controllo ricordano mnemonicamente i termini *left overlap* e *right overlap*. Tra le altre possibilità, queste istruzioni si prestano per consentire la sovrapposizione del testo; per esempio, si può sbarrare un lettera «o» con una barra obliqua inversa semplicemente così: '**o\llap{$\backslash$}**'.

La parola di controllo '**\null**' inserisce una scatola vuota equivalente all'istruzione '**\hbox{}**'; invece, '**\strut**' crea una scatola vuota alta e profonda quanto una parentesi tonda; nello stesso modo funziona la parola di controllo '**\mathstrut**' che invece si usa in modalità matematica.

Sempre in modalità matematica, si può usare '**\underline**' che sottolinea il testo contenuto nell'espressione:

\underline {*espressione*}

In questo caso, per fare in modo che la linea di sottolineatura sia sempre alla stessa altezza, si inserisce generalmente una scatola vuota generata da '**\mathstrut**'. L'esempio seguente riepiloga l'uso delle scatole orizzontali:

```
Paragrafo normale 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 \par
inizio paragrafo; \hbox {testo in scatola}; fine paragrafo.
\par
inizio paragrafo; \hbox to 5cm {testo in scatola}; fine
paragrafo. \par
inizio paragrafo; \hbox to 5cm {testo \hss in scatola}; fine
paragrafo. \par
\line{Riga di testo normale allineata a sinistra e a
destra.}
\line{\llap {a)} Bla bla bla \rlap {(b}}
$\underline {testo sottolineato}, g j p q y,
\underline {\mathstrut testo sottolineato con una linea
pi\grave u bassa}$ \par
```

Paragrafo normale 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
inizio paragrafo; testo in scatola; fine paragrafo.
inizio paragrafo; testo in scatola; fine paragrafo.
inizio paragrafo; testo in scatola; fine paragrafo.
Riga di testo normale allineata a sinistra e a destra.
a) Bla bla bla (b
$\underline{testosottolineato}, gjpqy, \underline{testosottolineatoconunalineapi\grave{u}bassa}$

L'esempio dell'utilizzo più importante delle scatole orizzontali è dato dalla macro '**\line**' che è definita semplicemente così:

```
\def\line{\hbox to\hsize}
```

In pratica, quando si scrive '**\line{*testo*}**', non si sta indicando un argomento alla macroistruzione, ma si ottiene solo la sostituzione di '**\line**' con '**\hbox to\hsize**', per cui alla fine si ottiene '**\hbox to\hsize {*testo*}**' complessivamente. In altri termini, la macroistruzione '**\line**' è seguita da un gruppo che, come tale, isola il testo che racchiude.

Per la realizzazione di scatole verticali sono disponibili tre parole di controllo: '**\vbox**', '**\vtop**' e '**\vcenter**', dove l'ultima interviene solo in ambienti matematici.

\vbox {*espressione*} | \vbox to *altezza* {*espressione*}

\vtop {*espressione*} | \vtop to *altezza* {*espressione*}

\vcenter {*espressione*} | \vcenter to *altezza* {*espressione*}

I due modelli sintattici alternativi, '***nome gruppo***' e '***nome* to *altezza gruppo***', consentono di creare scatole alte quanto serve, in base al contenuto, oppure alte esattamente quanto richiesto.

La scatola che si ottiene con '**\vbox**', colloca la posizione inferiore al livello della riga da cui si parte; estendendosi verso l'alto; la scatola che si ottiene con '**\vtop**' si estende invece verso il basso; infine, la scatola generata con '**\vcenter**' risulta centrata verticalmente rispetto alla posizione di partenza.

La larghezza di queste scatole è controllata dal valore restituito da '**\hsize**'; valore che può essere cambiato opportunamente al loro interno.

```
Paragrafo normale 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 \par
A: \vbox {\hsize=10cm Scatola verticale, larga 10 cm, che si
espande verso l'altro. 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} \par
B: \vtop {\hsize=10cm Scatola verticale, larga 10 cm, che si
espande verso il basso. 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} \par
C: $\vcenter {\hsize=10cm Scatola verticale, larga 10 cm,
che si espande verso l'altro e verso il basso. bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla}$ \par
D: \vbox to 3cm {\hsize=7cm Questa \'e una scatola
rettangolare, larga sette centimetri e alta tre centimetri.
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla}
\par
Paragrafo normale 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 \par
```

Paragrafo normale 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

Scatola verticale, larga 10 cm, che si espande verso l'altro. bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla

A: bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla

B: Scatola verticale, larga 10 cm, che si espande verso il basso. 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

Scatola verticale, larga 10 cm, che si espande verso l'altro e

C: verso il basso. 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

Paragrafo normale 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

Questa è una scatola rettangolare, larga sette centimetri e alta tre centimetri. bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla

D:

Paragrafo normale 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

L'esempio mostra quattro casi di scatole inserite in paragrafi di testo normale, dove ogni paragrafo inizia con una lettera alfabetica per dare un riferimento visivo alla collocazione della stessa. Si può osservare l'ultimo caso in cui si fissa anche la larghezza della scatola intervenendo nel valore a cui si accede con la parola di controllo '**\hsize**'.

Benché la parola di controllo '**\vcenter**' debba essere inserita in un ambiente matematico, il gruppo che controlla si trova in un ambiente normale.

49.14.1 Spostamento orizzontale delle scatole

Le due macro '**\moveleft**' e '**\moveright**', consentono di spostare orizzontalmente una scatola verticale od orizzontale che sia:

\moveleft ***lunghezza*** \hbox{***contenuto***} | \vbox{***contenuto***}
\moveright ***lunghezza*** \hbox{***contenuto***} | \vbox{***contenuto***}

Come si vede dal modello sintattico, si ottiene lo spostamento, rispettivamente, verso sinistra o verso destra, della lunghezza indicata. Si osservi l'esempio seguente:

```
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.\par
\moveright 3 cm \vbox{
\hsize=7cm
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.\par
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.\par
}\par
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.\par
```


In pratica, dopo un paragrafo normale, viene richiesto un rientro di 3 cm per una scatola verticale contenente del testo, nella quale la giustezza è di soli 7 cm. Infine, un altro paragrafo normale permette di verificare che al di fuori della scatola, la giustezza torna quella di prima.

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 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 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 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 bla bla bla bla.

Eventualmente, un piccolo spazio verticale può evitare che il testo che precede la scatola verticale si trovi a toccare il contenuto della stessa.

49.14.2 Scatole più complesse

Le scatole definite con '**\hbox**', '**\vbox**' e altre macroistruzioni simili, possono essere inserite una dentro l'altra per formare degli effetti speciali, soprattutto se si abbina questo fatto all'uso delle linee orizzontali e verticali. Di solito ci si affida a macroistruzioni già pronte, oppure a esempi documentati; comunque è utile cercare di comprendere cosa sta sotto. Viene proposto un esempio, attraverso il quale si crea una macroistruzione che racchiude del testo dentro una scatola orizzontale bordata:

```
\def\scatola#1{%
    \leavevmode                                  % 5
    \hbox{%                                      % 4
        \vrule
        \vtop{%                                  % 3
            \vbox{%                              % 2
                \hrule\kern 1pt
                \hbox{%
                    \kern 1pt \strut#1\kern 1pt  % 1
                }%
            }%
            \kern 1pt
            \hrule
        }%
        \vrule
    }
}
```

Questo esempio così articolato è tratto da *TeX made easy* di Daniel M. Zirin. Per riuscire a comprenderlo, occorre partire dal centro.

1. L'istruzione '**\kern 1pt \strut#1\kern 1pt**' inserisce il parametro della macro aggiungendo un punto di spazio a sinistra e a destra, garantendo di avere a disposizione lo spazio verticale necessario a scrivere qualunque lettera (che si espanda in alto e in basso, rispetto alla base).
2. Tutto questo è racchiuso in una scatola verticale che, prima disegna una linea orizzontale, quindi lascia un punto di spazio in verticale, infine inserisce l'istruzione precedente.
3. Il tutto è contenuto in un'altra scatola verticale che si espande verso il basso, dove alla fine si aggiunge un altro spazio verticale di un punto e un'altra linea orizzontale.
4. Ancora una volta, tutto è contenuto dentro un'altra scatola, questa volta orizzontale, che prima e dopo aggiunge due linee verticali.

5. Per sicurezza, la macroistruzione inizia chiamando '**\leavevmode**' per garantire il passaggio alla modalità orizzontale.

Ecco come si comporta la macroistruzione:

```
\scatola{Attenzione!} con le scatole ci si confonde
facilmente.
```

Attenzione! con le scatole ci si confonde facilmente.

Si osservi che nella definizione della macroistruzione '**\scatola**' sono stati inseriti molti commenti per consentire la suddivisione del codice su più righe, senza interferire con la composizione finale.

49.15 Tabelle

La gestione delle tabelle con TeX è piuttosto complessa. Nelle sezioni successive vengono mostrati soltanto i due tipi fondamentali di tabella di TeX; Si tenga presente che esistono altri modi di crearne, per esempio inserendo scatole dentro altre scatole, ma anche l'ambiente matematico dispone di un tipo di tabella che facilita la realizzazione di matrici.

49.15.1 Tabulazione

Le macchine da scrivere tradizionali disponevano di un sistema di stop di tabulazione, programmato di volta in volta, con il quale si potevano allineare i dati in colonne, formando così delle tabelle. Il sistema più semplice per creare tabelle con TeX è proprio quello che si rifà all'uso della macchina da scrivere, attraverso la definizione di stop di tabulazione che poi vengono richiamati quando desiderato. Si osservi l'esempio seguente:

```
\settabs 3 \columns

Testo normale, 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

\+ nero & 0 & zero \cr
\+ marrone & 1 & uno \cr
\+ rosso & 2 & due \cr
\+ arancio & 3 & tre \cr
\+ giallo & 4 & quattro \cr
\+ verde & 5 & cinque \cr
\+ blu & 6 & sei \cr
\+ viola & 7 & sette \cr
\+ grigio & 8 & otto \cr
\+ bianco & 9 & nove \cr

Testo normale, 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

\+ argento & 10 \% \cr
\+ oro & 5 \% \cr
\+ & 2 \% \cr
\+ & 1 \% \cr
```

L'istruzione '**\settabs 3 \columns**' definisce, da quel punto in poi, l'inserimento di due stop di tabulazione, dividendo la giustezza del testo in tre parti uguali. Dopo tale definizione, si può scrivere normalmente, come se nulla fosse cambiato, oppure si può fare riferimento esplicito alla tabulazione, iniziando e concludendo le righe con '**\+**' e con '**\cr**'. Nello spazio tra queste due istruzioni, la e-commerciale ('**&**') ha il significato di carattere di tabulazione, con il quale si raggiunge lo stop successivo. Si può vedere il risultato della composizione dell'esempio nella figura successiva.

Testo normale, 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

nero	0	zero
marrone	1	uno
rosso	2	due
arancio	3	tre
giallo	4	quattro
verde	5	cinque
blu	6	sei
viola	7	sette
grigio	8	otto
bianco	9	nove

Testo normale, 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

argento	10 %
oro	5 %
	2 %
	1 %

Come si dovrebbe notare leggendo l'esempio, non è necessario usare tutte le colonne previste, così come è possibile saltare le colonne che non interessano. Quello che non si intuisce, invece, è la possibilità di fare riferimento a stop di tabulazione oltre alla quantità dichiarata inizialmente: in tal caso si raggiungono colonne successive che, ovviamente, vanno oltre la giustezza del testo.

L'istruzione del tipo '**\settabs *n* \columns**' ha effetto a partire dal punto in cui viene utilizzata. Se si vuole cambiare tabulazione, basta inserire una nuova istruzione di questo tipo. Inoltre, è possibile usare anche i raggruppamenti per cambiare temporaneamente la tabulazione di una tabella; tutto funziona in modo molto intuitivo.

```
\settabs 3 \columns
\+ nero & 0 & zero \cr
\+ marrone & 1 & uno \cr
\+ rosso & 2 & due \cr
{\settabs 4 \columns
\+ arancio & 3 & tre \cr
\+ giallo & 4 & quattro \cr
\+ verde & 5 & cinque \cr
\+ blu & 6 & sei \cr
}
\+ viola & 7 & sette \cr
\+ grigio & 8 & otto \cr
\+ bianco & 9 & nove \cr
```

In questa variante semplificata dell'esempio già presentato, le righe dall'arancio al blu hanno una tabulazione più stretta, perché si divide la giustezza in quattro porzioni.

nero	0		zero
marrone	1		uno
rosso	2		due
arancio	3	tre	
giallo	4	quattro	
verde	5	cinque	
blu	6	sei	
viola	7		sette
grigio	8		otto
bianco	9		nove

Gli stop di tabulazione possono essere collocati in modo diverso, indicando una riga di esempio al posto della definizione pura e semplice della quantità di colonne che si vogliono ottenere. In pratica, si hanno a disposizione due istruzioni alternative:

```
\settabs n \columns
```

```
\settabs \+ colonna_1 [& colonna_n]… \cr
```

Viene riproposto l'esempio già visto, in forma ancora più breve, in cui viene inserito lo stesso modello usato per definire gli stop di tabulazione alla fine della tabella, per poter vedere effettivamente in che modo vengono misurati gli spazi:

```
\settabs \+ MMMMMMMMMMM & MMMMM & MMMMMMM \cr
\+ nero & 0 & zero \cr
\+ marrone & 1 & uno \cr
\+ rosso & 2 & due \cr
\+ arancio & 3 & tre \cr
Il modello usato per gli stop \‘e il seguente:
```

```
\+ MMMMMMMMMMMMM & MMMMM & MMMMMMM \cr
```

nero	0	zero
marrone	1	uno
rosso	2	due
arancio	3	tre

Il modello usato per gli stop è il seguente:
MMMMMMMMMMMMM MMMMM MMMMMMM

Naturalmente, il modello delle colonne può essere anche più simile al testo usato realmente nella tabella. In questo caso, con la lettera «M» maiuscola, si fa riferimento a un quadratone in modo semplice e chiaro.

Il testo che si inserisce all'interno di uno spazio che rappresenta una cella, esclusa l'ultima di ogni riga, è racchiuso implicitamente in un gruppo che si comporta come la macro '**`\leftline`**'. Questo fatto consente quindi di gestire gli allineamenti nell'ambito delle celle; inoltre, garantisce che il testo di ognuna di queste celle sia isolato in un gruppo autonomo.

```
\settabs
\+ MMMMMMMMMMMMM & MMMMM        & MMMMMMM      & \cr
\+ \hrulefill   & \dotfill     & \hrulefill   & \cr
\+ \hfil Colore & \hfil Cifra  & \hfil Cifra  & \cr
\+ \hrulefill   & \dotfill     & \hrulefill   & \cr
\+ nero         & \hfill 0     & \hfil zero   & \cr
\+ marrone      & \hfill 1     & \hfil uno    & \cr
\+ \bf rosso    & \hfill 2     & \hfil due    & \cr
\+ arancio      & \hfill 3     & \hfil tre    & \cr
\+ \hrulefill   & \dotfill     & \hrulefill   & \cr
```

L'esempio mostra una tabella, in cui il sorgente è organizzato in modo da facilitarne la lettura. In particolare, il modello per gli stop di tabulazione è stato messo nella riga successiva a quella in cui appare '**`\settabs`**'; inoltre, c'è una cella finale in più che non viene usata. Infatti, dalle celle in cui il testo si comporta come se fosse usata la macro '**`\leftline`**' è esclusa l'ultima che precede l'istruzione '**`\cr`**'.

Colore	Cifra	Cifra
nero	0	zero
marrone	1	uno
rosso	2	due
arancio	3	tre

Bisogna ricordare che la macro '**`\leftline`**' include uno spazio leggermente elastico alla fine, corrispondente a '**`\hfil`**'. In tal modo, per centrare il testo in una cella, basta farlo precedere da un altro '**`\hfil`**', se invece si vuole ottenere l'allineamento a destra, si deve usare inizialmente uno spazio più elastico come '**`\hfill`**'.

Nell'esempio mostrato si riesce ad arrivare a una tabella abbastanza «dignitosa», con tanto di intestazione, delimitata da linee orizzontali. Naturalmente, si potrebbero usare linee dello stesso tipo, ma in questo caso si vuole mostrare bene lo spazio di competenza di ogni colonna. Al posto di '**`\hrulefill`**' o di '**`\dotfill`**' si può anche usare '**`\hrule`**', al di fuori delle celle, nel modo seguente:

```
\settabs
\+ MMMMMMMMMMMMM & MMMMM        & MMMMMMM      & \cr
\hrule
\+ \hfil Colore & \hfil Cifra  & \hfil Cifra  & \cr
\hrule
\+ nero         & \hfill 0     & \hfil zero   & \cr
\+ marrone      & \hfill 1     & \hfil uno    & \cr
\+ \bf rosso    & \hfill 2     & \hfil due    & \cr
\+ arancio      & \hfill 3     & \hfil tre    & \cr
\hrule
```

Tuttavia, in tal caso si vede che le linee orizzontali che si ottengono toccano il testo delle righe della tabella:

Colore	Cifra	Cifra
nero	0	zero
marrone	1	uno
rosso	2	due
arancio	3	tre

Per risolvere il problema, può essere conveniente l'uso della macro '**`\strut`**', con cui si dà alle righe l'altezza e la profondità corrispondente a quella di una parentesi tonda, senza che questa appaia effettivamente nel testo:

```
\settabs
\+ MMMMMMMMMMMMM        & MMMMM        & MMMMMMM      & \cr
\hrule
\+ \strut \hfil Colore  & \hfil Cifra  & \hfil Cifra  & \cr
\hrule
\+ \strut nero          & \hfill 0     & \hfil zero   & \cr
\+ \strut marrone       & \hfill 1     & \hfil uno    & \cr
\+ \strut \bf rosso     & \hfill 2     & \hfil due    & \cr
\+ \strut arancio       & \hfill 3     & \hfil tre    & \cr
\hrule
```

In tal modo si ottiene anche il risultato di avere tutte le righe della tabella della stessa altezza, quella massima, anche se il testo che contengono non lo richiederebbe.

Colore	Cifra	Cifra
nero	0	zero
marrone	1	uno
rosso	2	due
arancio	3	tre

Se si inserisce una tabella del genere in una scatola verticale, questa scatola può essere spostata facilmente in modo orizzontale, con l'istruzione '**`\moveright`**', come nell'esempio seguente, in cui la si sposta di 3 cm:

```
Paragrafo di esempio, 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.

\moveright 3 cm \vbox{
\settabs
\+ MMMMMMMMMMMMM        & MMMMM        & MMMMMMM      & \cr
\hrule
\+ \strut \hfil Colore  & \hfil Cifra  & \hfil Cifra  & \cr
\hrule
\+ \strut nero          & \hfill 0     & \hfil zero   & \cr
\+ \strut marrone       & \hfill 1     & \hfil uno    & \cr
\+ \strut \bf rosso     & \hfill 2     & \hfil due    & \cr
\+ \strut arancio       & \hfill 3     & \hfil tre    & \cr
\hrule
}
```

Tuttavia, come si vede sotto, in questo modo il testo che precede la tabella si trova a toccare la tabella stessa:

Paragrafo di esempio, 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.

Colore	Cifra	Cifra
nero	0	zero
marrone	1	uno
rosso	2	due
arancio	3	tre

Per rimediare basta aggiungere un piccolo spazio verticale, per esempio con l'istruzione '**`\vskip`**', prima della tabella.

Volendo arrivare a centrare la tabella, si può inserire la scatola verticale nella macro '**`\line`**', come nell'esempio seguente:

```
Paragrafo di esempio, 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.\par
\vskip 1 em
\line{\hfil\vbox{
\settabs
\+ MMMMMMMMMMMMM        & MMMMM        & MMMMMMM      & \cr
\hrule
\+ \strut \hfil Colore  & \hfil Cifra  & \hfil Cifra  & \cr
\hrule
\+ \strut nero          & \hfill 0     & \hfil zero   & \cr
\+ \strut marrone       & \hfill 1     & \hfil uno    & \cr
\+ \strut \bf rosso     & \hfill 2     & \hfil due    & \cr
\+ \strut arancio       & \hfill 3     & \hfil tre    & \cr
\hrule
}\hfil}
```

Si osservi l'uso dell'istruzione '**`\vskip`**' per ottenere una piccola

distanza tra il testo e la tabella, come già descritto in precedenza:

Paragrafo di esempio, 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.

Colore	Cifra	Cifra
nero	0	zero
marrone	1	uno
rosso	2	due
arancio	3	tre

49.15.2 Tabelle più complesse

Attraverso l'istruzione '**`\halign`**' è possibile definire delle tabelle un po' più complesse rispetto al sistema di stop di tabulazione già presentato nella sezione precedente. L'istruzione in questione viene utilizzata sinteticamente nel modo seguente:

```
\halign{modello_delle_righe
riga
...
}
```

In particolare, il modello delle righe utilizza il simbolo '`#`' per fare riferimento al contenuto delle celle, mentre le righe hanno la forma consueta, con la differenza che manca il simbolo di controllo '`\+`' iniziale. L'esempio seguente è molto simile all'ultima tabella realizzata con l'uso degli stop di tabulazione, con tanto di centratura orizzontale:

```
Paragrafo di esempio, 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.\par
\vskip 1 em
\line{\hfil\vbox{
\halign{
\strut #        & \hfil # & \hfil # \hfil \cr
\noalign{\hrule}
Colore          & Cifra & Cifra \cr
\noalign{\hrule}
nero            & 0     & zero  \cr
marrone         & 1     & uno   \cr
\bf rosso       & 2     & due   \cr
arancio         & 3     & tre   \cr
\noalign{\hrule}
}
}\hfil}
```

La prima cosa che si deve osservare è il fatto che '**`\strut`**' viene usato solo nella dichiarazione del modello delle righe; ciò è sufficiente perché sia inserito automaticamente all'inizio di tutte le righe della tabella. Si può osservare anche che non è necessario definire la larghezza delle colonne nel modello, perché questa viene determinata automaticamente in base al contenuto delle righe stesse. Infine, l'ultima colonna della tabella non si comporta diversamente dalle altre, pertanto non è più necessario il trucco della colonna finale vuota.

Paragrafo di esempio, 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.

Colore	Cifra	Cifra
nero	0	zero
marrone	1	uno
rosso	2	due
arancio	3	tre

A differenza delle tabelle realizzate per mezzo della tabulazione, quando si vuole inserire una linea orizzontale con l'istruzione '**`\hrule`**', occorre inserirla nell'istruzione '**`\noalign`**'. In pratica, con '**`\noalign`**' si può inserire qualcosa che riguarda tutta la riga, senza suddivisione in colonne; tuttavia, è meglio evitare di inserire testo normale, perché in tal caso verrebbe perso l'effetto dell'allineamento orizzontale introdotto con '**`\line`**' e '**`\vbox`**'.

È importante comprendere che ciò che si inserisce nel modello iniziale delle righe, viene inserito nello stesso modo all'interno delle righe. Si osservi la variante seguente:


```
\line{\hfil\vbox{
\halign{
\strut \quad \bf # & \hfil #          & \hfil (#) \hfil \cr
\noalign{\hrule}
Colore             & \bf Cifra \hfil  & \bf Cifra     \cr
\noalign{\hrule}
nero               & 0                & zero          \cr
marrone            & 1                & uno           \cr
rosso              & 2                & due           \cr
arancio            & 3                & tre           \cr
\noalign{\hrule}
}
}\hfil}
```

Per prima cosa, è stato aggiunto un piccolo spazio orizzontale, di un quadratone, nella prima cella, attraverso la macro '**`\quad`**'; in tal modo si evita che la prima colonna inizi esattamente sul bordo sinistro, per motivi estetici. Inoltre, si fa in modo che tutta la prima colonna appaia in neretto, senza dover intervenire in ogni cella. Nel modello dell'ultima colonna, si racchiude il riferimento alle celle tra parentesi tonde, solo a scopo dimostrativo (infatti il risultato che si ottiene non è perfetto esteticamente). Dal momento che si vuole la riga di intestazione tutta in neretto, si interviene singolarmente nella seconda e nella terza cella di questa riga; in particolare, la parola «Cifra» della seconda colonna, viene centrata, aggiungendo un '**`\hfil`**' finale, che contrasta e bilancia quanto dichiarato nel modello corrispondente.

Colore	**Cifra**	(**Cifra**)
nero	0	(zero)
marrone	1	(uno)
rosso	2	(due)
arancio	3	(tre)

Dal momento che tutto ciò che si inserisce nella riga del modello viene messo anche nelle righe della tabella, per ottenere delle linee verticali si possono usare delle istruzioni '**`\vrule`**':

```
\line{\hfil\vbox{
\halign{
\strut\vrule \quad \bf # & \vrule \hfil #  & \vrule \hfil (#) \hfil \vrule \cr
\noalign{\hrule}
Colore                   & \bf Cifra \hfil & \bf Cifra                    \cr
\noalign{\hrule}
nero                     & 0               & zero                         \cr
marrone                  & 1               & uno                          \cr
rosso                    & 2               & due                          \cr
arancio                  & 3               & tre                          \cr
\noalign{\hrule}
}
}\hfil}
```

In questo caso il risultato estetico non è ancora perfetto, ma ormai dovrebbe essere chiaro come si può intervenire nel modello delle righe.

Colore	**Cifra**	(**Cifra**)
nero	0	(zero)
marrone	1	(uno)
rosso	2	(due)
arancio	3	(tre)

Esiste la possibilità di unire assieme più celle nella stessa riga, attraverso l'uso della macroistruzione '**`\multispan{`*`n`*`}`**', dove ***n*** rappresenta la quantità di colonne da unire. Tuttavia, in tal caso non viene preso in considerazione il formato stabilito nella riga di intestazione. Si osservi l'esempio seguente:

```
\line{\hfil\vbox{
\halign{
\strut\vrule \quad \bf # & \vrule \hfil #  & \vrule \hfil (#) \hfil \vrule \cr
\noalign{\hrule}
Colore                   & \bf Cifra \hfil & \bf Cifra                    \cr
\noalign{\hrule}
nero                     & 0               & zero                         \cr
marrone                  & 1               & uno                          \cr
\multispan{2} rosso                        & due                          \cr
arancio                  & 3               & tre                          \cr
\noalign{\hrule}
}
}\hfil}
```

Come si vede sotto, il contenuto della cella allargata appare centrato nello spazio che si ritrova ad avere a disposizione, ma ciò è ottenuto con uno spazio leggermente elastico, contrastabile facilmente.

Colore	Cifra	(Cifra)
nero	0	(zero)
marrone	1	(uno)
rosso		(due)
arancio	3	(tre)

Per sistemare l'allineamento, ripristinare il neretto e la linea verticale alla sinistra, basta intervenire nella cella; in particolare è sufficiente uno spazio con una piccola elasticità per contrastare l'allineamento normale al centro:

```
\line{\hfil\vbox{
\halign{
\strut\vrule \quad \bf # & \vrule \hfil #  & \vrule \hfil (#) \hfil \vrule \cr
\noalign{\hrule}
Colore                   & \bf Cifra \hfil & \bf Cifra                    \cr
\noalign{\hrule}
nero                     & 0               & zero                         \cr
marrone                  & 1               & uno                          \cr
\multispan{2}\vrule\quad\bf rosso scuro \hfil    & due                     \cr
arancio                  & 3               & tre                          \cr
\noalign{\hrule}
}
}\hfil}
```

Colore	Cifra	(Cifra)
nero	0	(zero)
marrone	1	(uno)
rosso scuro		(due)
arancio	3	(tre)

Per evitare che in una cella venga inserito tutto ciò che appare nel modello delle righe, basta usare l'istruzione '**\omit**'. Si osservi l'esempio, in cui si risolve il problema delle parentesi tonde attorno al titolo dell'ultima colonna:

```
\line{\hfil\vbox{
\halign{
\strut\vrule \quad \bf # & \vrule \hfil #  & \vrule \hfil (#) \hfil \vrule \cr
\noalign{\hrule}
Colore                &\bf Cifra\hfil &\omit\vrule\hfil\bf Cifra\hfil\vrule\cr
\noalign{\hrule}
nero                     & 0               & zero                         \cr
marrone                  & 1               & uno                          \cr
rosso                    & 2               & due                          \cr
arancio                  & 3               & tre                          \cr
\noalign{\hrule}
}
}\hfil}
```

Colore	Cifra	Cifra
nero	0	(zero)
marrone	1	(uno)
rosso	2	(due)
arancio	3	(tre)

Prima di concludere l'argomento, occorre fare presente la possibilità che TeX allunghi il testo verticalmente in fase di composizione. Se ciò accade nell'ambito di una tabella nella quale si utilizza '**\vrule**' per ottenere delle linee verticali, si rischia di vedere queste linee spezzettate. Per evitare che nell'ambito della tabella TeX possa fare degli allungamenti, si può usare la macro '**\offinterlineskip**' dentro la scatola verticale:

```
\line{\hfil\vbox{\offinterlineskip
\halign{
\strut\vrule \quad \bf # & \vrule \hfil #  & \vrule \hfil (#) \hfil \vrule \cr
\noalign{\hrule}
Colore                &\bf Cifra\hfil &\omit\vrule\hfil\bf Cifra\hfil\vrule\cr
\noalign{\hrule}
nero                     & 0               & zero                         \cr
marrone                  & 1               & uno                          \cr
rosso                    & 2               & due                          \cr
arancio                  & 3               & tre                          \cr
\noalign{\hrule}
}
}\hfil}
```

49.16 Ambienti matematici

Come già accennato, TeX distingue tra due modalità di funzionamento: un contesto normale e un contesto matematico. L'ambiente matematico si introduce e si conclude con il simbolo '**$**' e in tale situazione diventano disponibili delle istruzioni che non si possono utilizzare al di fuori di questo ambito, mentre alcune istruzioni dell'ambiente normale non lo sono più.

Anche se nelle sezioni successive si fa riferimento soltanto all'ambiente matematico, nelle tabelle riassuntive mostrate, le sequenze di controllo che possono essere usate solo nell'ambiente matematico appaiono delimitate sempre tra una coppia di '**$**', in modo da evidenziare il contesto del loro utilizzo ed evitare confusione.

49.16.1 Due situazioni differenti

Esistono due modi di mostrare un'informazione matematica: all'interno di testo normale, oppure in un blocco a parte. Quando si inserisce l'informazione dentro del testo normale, si delimita l'ambiente matematico tra due simboli '**$**', per esempio nel modo seguente:

```
Se $R$ \'e il ritardo di ogni singola consegna, $T$
\'e il tempo a disposizione per la consegna, $I$ \'e il
valore della merce, il ritardo medio si esprime come
${\sum {R \over T} \cdot I}\over{\sum I}$.
```

In tal caso, come si vede dal risultato della composizione, il testo ne risente, in particolare per lo spazio tra le righe che può essere aumentato.

Se R è il ritardo di ogni singola consegna, T è il tempo a disposizione per la consegna, I è il valore della merce, il ritardo medio si esprime come $\frac{\sum \frac{R}{T} \cdot I}{\sum I}$.

In alternativa, le formule possono essere messe in un blocco separato, come nell'esempio seguente:

```
Se $R$ \'e il ritardo di ogni singola consegna, $T$ \'e il
tempo a disposizione per la consegna, $I$ \'e il valore
della merce, il ritardo medio si esprime secondo la formula
seguente:\par
$${\sum {R \over T} \cdot I}\over{\sum I}$$
```

L'effetto è quello di ottenerle al centro della giustezza, con una spaziatura verticale adeguata rispetto al testo che precede e che segue.

Se R è il ritardo di ogni singola consegna, T è il tempo a disposizione per la consegna, I è il valore della merce, il ritardo medio si esprime secondo la formula seguente:

$$\frac{\sum \frac{R}{T} \cdot I}{\sum I}$$

Negli esempi è stato mostrato l'uso dell'ambiente matematico anche per delimitare i nomi delle variabili. Ciò permette di mantenere coerenza con la forma mostrata nelle formule.

Ciò che viene delimitato in una coppia di '**$**' o di '**$$**', non può essere suddiviso in «righe» differenti (tranne il caso di forme tabellari particolari, come le matrici o i sistemi di equazioni), al contrario di quanto avviene per la composizione del testo normale; inoltre, il sorgente non può contenere righe vuote o bianche. In pratica si tratta di un blocco orizzontale compatto e indivisibile.

49.17 Spazi orizzontali

Si osservi subito l'esempio seguente, in cui si scrive la parola «affittare» all'interno di un ambiente matematico, prima in modo normale, quindi spaziando le lettere:

```
$affittare$\par
$a    f    f    i    t  t  a  r  e$\par
```

Come si può osservare dalla composizione che si ottiene, il risultato è esattamente lo stesso. In pratica non viene più preso in considerazione il legato tipografico e gli spazi sono ignorati completamente:

$$affittare$$
$$a\ f\ f\ i\ t\ t\ a\ r\ e$$

Per inserire degli spazi in un ambiente matematico, l'unico modo è quello di usare sequenze di controllo specifiche, che possano essere accettate anche in questo ambiente. La tabella 49.245 riepiloga le sequenze di controllo che possono essere usate in un ambiente matematico per ottenere degli spazi orizzontali. Alcune di queste sequenze sono valide solo in un ambito matematico, pertanto appaiono circoscritte da una coppia di '**$**', in modo da sottolineare questa loro caratteristica.

Tabella 49.245. Sequenze di controllo per ottenere delle spaziature orizzontali in ambito matematico.

Sequenza	Significato
`\qquad`	Due quadratoni.
`\quad`	Un quadratone.
`\<SP>`	Uno spazio «normale».
`$\;$`	Uno spazio spesso (5/8 di quadratone).
`$\>$`	Uno spazio medio (2/9 di quadratone).
`$\.$`	Uno spazio molto sottile (1/6 di quadratone).
`$\!$`	Uno spazio negativo (-1/6 di quadratone).

Per inserire del testo descrittivo all'interno di un ambiente matematico, lo si può delimitare in una scatola orizzontale, con l'istruzione '**`\hbox`**'. Si osservi l'esempio:

```
$prima durante e dopo$\par
$prima \hbox{ durante } e dopo$\par
```

In questo caso, si vuole isolare e spaziare la parola «durante» rispetto al resto, mostrandola con caratteri normali.

$$primaduranteedopo$$
$$prima\ \text{durante}\ edopo$$

49.17.1 Caratteri e simboli

Nell'ambiente matematico, il testo normale viene composto utilizzando un insieme di caratteri differente rispetto a quello normale, in cui anche la forma è particolare, essendo un tipo speciale di corsivo (si ottiene eventualmente con la macro '**`\mit`**'). Inoltre, sono disponibili dei simboli aggiuntivi, in particolare le lettere greche e altri simboli utili in matematica.

Tra le lettere greche, alcune si ottengono come caratteri dell'insieme normale. Nelle tabelle, quando si vuole sottolineare il fatto che si tratta di lettere scritte utilizzando il carattere tondo normale, si mostra un'istruzione del tipo '**`\hbox{\rm x}`**', che rappresenta il codice necessario all'inserimento in un ambiente matematico.

Tabella 49.248. Accenti nell'ambiente matematico.

Simbolo	Codice	Annotazioni
$\acute{o}$	$\acute o$	Accento acuto matematico.
$\grave{o}$	$\grave o$	Accento grave matematico.
$\hat{o}$	$\hat o$	Accento circonflesso matematico.
$\ddot{o}$	$\ddot o$	
$\tilde{o}$	$\tilde o$	
$\bar{o}$	$\bar o$	
$\dot{o}$	$\dot o$	
$\breve{o}$	$\breve o$	
$\check{o}$	$\check o$	
$\vec{o}$	$\vec o$	
$\imath$	$\imath$	Da usare per aggiungere un accento.
$\jmath$	$\jmath$	Da usare per aggiungere un accento.
$\widehat{abc}$	$\widehat{abc}$	
$\widetilde{abc}$	$\widetilde{abc}$	

Tabella 49.249. Lettere greche.

Simbolo	Codice	Simbolo	Codice	Simbolo	Codice	Nome
α	α			A	\hbox{\rm A}	Alfa
β	β			B	\hbox{\rm B}	Beta
γ	γ			Γ	Γ	Gamma
δ	δ			Δ	Δ	Delta
ϵ	ϵ	ε	ε	E	\hbox{\rm E}	Epsilon
ζ	ζ			Z	\hbox{\rm Z}	Zeta
η	η			H	\hbox{\rm H}	Eta
θ	θ	ϑ	ϑ	Θ	Θ	Theta
ι	ι			I	\hbox{\rm I}	Iota
κ	κ			K	\hbox{\rm K}	Kappa
λ	λ			Λ	Λ	Lambda
μ	μ			M	\hbox{\rm M}	Mu
ν	ν			N	\hbox{\rm N}	Nu
ξ	ξ			Ξ	Ξ	Xi
o	\hbox{\rm o}	o	o	O	\hbox{\rm O}	Omicron
π	π	ϖ	ϖ	Π	Π	Pi
ρ	ρ	ϱ	ϱ	P	\hbox{\rm P}	Rho
σ	σ	ς	ς	Σ	Σ	Sigma
τ	τ			T	\hbox{\rm T}	Tau
υ	υ			Υ	Υ	Upsilon
ϕ	ϕ	φ	φ	Φ	Φ	Phi
χ	χ			X	\hbox{\rm X}	Chi
ψ	ψ			Ψ	Ψ	Psi
ω	ω			Ω	Ω	Omega

Tabella 49.250. Simboli matematici comuni.

Simbolo	Codice	Annotazioni
$\backslash$	$\backslash$	Barra obliqua inversa.
$\aleph$	$\aleph$	
$\hbar$	$\hbar$	
ℓ	ℓ	
$\wp$	$\wp$	
$\Re$	$\Re$	
$\Im$	$\Im$	
∂	∂	Derivata parziale.
∞	∞	Infinito.
$\prime$	$\prime$	Primo.
$'$	$'
$\emptyset$	$\emptyset$	
∇	∇	
$\top$	$\top$	
$\bot$	$\bot$	
$\vert$	$\vert$	
$\vert$	$\vert$	Equivalente a $|$.
$\Vert$	$\Vert$	
$\Vert$	$\Vert$	Equivalente a $\|$.
$\angle$	$\angle$	
$\triangle$	$\triangle$	
$\forall$	$\forall$	
$\exists$	$\exists$	
$\neg$	$\neg$	
$\flat$	$\flat$	
$\natural$	$\natural$	
$\sharp$	$\sharp$	
$\surd$	$\surd$	Radice senza la linea superiore.
$\clubsuit$	$\clubsuit$	Fiori.
$\diamondsuit$	$\diamondsuit$	Quadri.
$\heartsuit$	$\heartsuit$	Cuori.
$\spadesuit$	$\spadesuit$	Picche.

Tabella 49.251. Operatori binari comuni.

Simbolo	Codice	Annotazioni
$\pm$	$\pm$	Più o meno.
$\mp$	$\mp$	Meno o più.
$\cdot$	$\cdot$	
$\setminus$	$\setminus$	
$+$	$+$	Somma.
$-$	$-$	Sottrazione.
$\times$	$\times$	Moltiplicazione.
$\div$	$\div$	Divisione.
	$*$	Moltiplicazione discrezionale.
$\ast$	$\ast$	Asterisco.
$\star$	$\star$	Stella.
$\diamond$	$\diamond$	
$\circ$	$\circ$	
$\bullet$	$\bullet$	
$\cap$	$\cap$	Intersezione.
$\cup$	$\cup$	Unione.
$\uplus$	$\uplus$	
$\sqcap$	$\sqcap$	
$\sqcup$	$\sqcup$	
$\triangleleft$	$\triangleleft$	
$\triangleright$	$\triangleright$	
$\wr$	$\wr$	
$\bigcirc$	$\bigcirc$	
$\bigtriangleup$	$\bigtriangleup$	
$\bigtriangledown$	$\bigtriangledown$	
$\vee$	$\vee$	
$\vee$	$\lor$	OR logico (uguale a $\vee$).
$\wedge$	$\wedge$	
$\wedge$	$\land$	AND logico (uguale a $\wedge$).
$\oplus$	$\oplus$	
$\ominus$	$\ominus$	
$\otimes$	$\otimes$	
$\oslash$	$\oslash$	
$\odot$	$\odot$	
$\dagger$	$\dagger$	
$\ddagger$	$\ddagger$	
II	$\amalg$	

Tabella 49.252. Operatori di relazione.

Simbolo	Codice	Simbolo	Codice
$=$	$=$	$\neq$	$\not =$
		$\neq$	$\neq$
		$\neq$	$\ne$
$\equiv$	$\equiv$	$\not\equiv$	$\not \equiv$
$\sim$	$\sim$	$\not\sim$	$\not \sim$
$\simeq$	$\simeq$	$\not\simeq$	$\not \simeq$
$\approx$	$\approx$	$\not\approx$	$\not \approx$
$\cong$	$\cong$	$\not\cong$	$\not \cong$
$\asymp$	$\asymp$	$\not\asymp$	$\not \asymp$
$\bowtie$	$\bowtie$		
$\doteq$	$\doteq$		
$\parallel$	$\parallel$		
$\perp$	$\perp$	$\not\perp$	$\not \perp$
$<$	$<$	$\not<$	$\not <$
$\leq$	$\leq$	$\not\leq$	$\not \leq$
$\le$	$\le$	$\not\le$	$\not \le$
$>$	$>$	$\not>$	$\not >$
$\geq$	$\geq$	$\not\geq$	$\not \geq$
$\ge$	$\ge$	$\not\ge$	$\not \ge$
$\prec$	$\prec$	$\not\prec$	$\not \prec$
$\preceq$	$\preceq$	$\not\preceq$	$\not \preceq$
$\succ$	$\succ$	$\not\succ$	$\not \succ$
$\succeq$	$\succeq$	$\not\succeq$	$\not \succeq$
$\ll$	$\ll$		
$\gg$	$\gg$		
$\subset$	$\subset$	$\not\subset$	$\not \subset$
$\subseteq$	$\subseteq$	$\not\subseteq$	$\not \subseteq$
$\supset$	$\supset$	$\not\supset$	$\not \supset$
$\supseteq$	$\supseteq$	$\not\supseteq$	$\not \supseteq$
$\sqsubseteq$	$\sqsubseteq$	$\not\sqsubseteq$	$\not \sqsubseteq$
$\sqsupseteq$	$\sqsupseteq$	$\not\sqsupseteq$	$\not \sqsupseteq$
$\mid$	$\mid$		
$\smile$	$\smile$		
$\frown$	$\frown$		
$\in$	$\in$	$\notin$	$\not \in$
$\ni$	$\ni$	$\not\ni$	$\not \ni$
$\owns$	$\owns$	$\not\owns$	$\not \owns$
$\vdash$	$\vdash$		
$\dashv$	$\dashv$		
$\models$	$\models$		
$\propto$	$\propto$		

Tabella 49.253. Frecce.

Simbolo	Codice	Simbolo	Codice
$\leftarrow$	$\leftarrow$	$\rightarrow$	$\rightarrow$
$\gets$	$\gets$	$\to$	$\to$
$\Leftarrow$	$\Leftarrow$	$\Rightarrow$	$\Rightarrow$
$\longleftarrow$	$\longleftarrow$	$\longrightarrow$	$\longrightarrow$
$\Longleftarrow$	$\Longleftarrow$	$\Longrightarrow$	$\Longrightarrow$
$\hookleftarrow$	$\hookleftarrow$	$\hookrightarrow$	$\hookrightarrow$
$\leftharpoonup$	$\leftharpoonup$	$\rightharpoonup$	$\rightharpoonup$
$\leftharpoondown$	$\leftharpoondown$	$\rightharpoondown$	$\rightharpoondown$
$\rightleftharpoons$	$\rightleftharpoons$		
$\iff$	$\iff$		
$\mapsto$	$\mapsto$		
$\longmapsto$	$\longmapsto$		
$\leftrightarrow$	$\leftrightarrow$	$\Leftrightarrow$	$\Leftrightarrow$
$\longleftrightarrow$	$\longleftrightarrow$	$\Longleftrightarrow$	$\Longleftrightarrow$
$\uparrow$	$\uparrow$	$\downarrow$	$\downarrow$
$\Uparrow$	$\Uparrow$	$\Downarrow$	$\Downarrow$
$\updownarrow$	$\updownarrow$	$\Updownarrow$	$\Updownarrow$
$\nwarrow$	$\nwarrow$	$\nearrow$	$\nearrow$
$\swarrow$	$\swarrow$	$\searrow$	$\searrow$
$\overleftarrow{ab}$	$\overleftarrow {ab}$	$\overrightarrow{ab}$	$\overrightarrow {ab}$
$\vec{ab}$	$\vec {ab}$		

Tabella 49.254. Ellissi.

Simbolo	Codice	Annotazioni
$\cdots$	$\cdots$	Ellissi orizzontale al centro.
$\ldots$	$\ldots$	Ellissi orizzontale in basso.
$\vdots$	$\vdots$	Ellissi verticale.
$\ddots$	$\ddots$	Ellissi diagonale.

Tabella 49.255. Altri simboli matematici.

Simbolo	Codice	Annotazioni
$\sum$	$\sum$	Sommatoria.
$\prod$	$\prod$	
$\coprod$	$\coprod$	
$\int$	$\int$	Integrale.
$\oint$	$\oint$	
$\bigcap$	$\bigcap$	Intersezione.
$\bigcup$	$\bigcup$	Unione.
$\bigsqcup$	$\bigsqcup$	
$\bigvee$	$\bigvee$	
$\bigwedge$	$\bigwedge$	
$\bigoplus$	$\bigoplus$	
$\biguplus$	$\biguplus$	
$\bigotimes$	$\bigotimes$	
$\bigodot$	$\bigodot$	

Eventualmente, è possibile modificare in modo sistematico il tipo di carattere da usare negli ambienti matematici. Si usano per questo due istruzioni alternative:

```
\everymath={macro_alternativa}
```

```
\everydisplay={macro_alternativa}
```

La prima si riferisce agli ambienti matematici inclusi nel testo normale, mentre la seconda riguarda gli ambienti che creano un blocco separato dal testo.

La macro alternativa a cui si fa riferimento può essere quella che definisce il tipo di carattere da usare. Si osservi l'esempio seguente:

```
\everymath={\bf}
\everydisplay={\rm}
la funzione $f(x)$ bla bla bla:
$$y = {{x^2}\over 2}$$
```

Si può vedere nel risultato seguente che nel testo normale si ottiene, mentre nel blocco separato si ha un tondo normale:

la funzione $\mathbf{f(x)}$ bla bla bla:

$$y = \frac{x^2}{2}$$

49.17.2 Dimensione del testo matematico

Il testo matematico, ovvero i simboli che si usano per le espressioni matematiche, viene dimensionato in modo automatico, salvo l'uso diretto di alcune istruzioni apposite. Le istruzioni sono nell'ordine: '**`\displaystyle`**', '**`\textstyle`**', '**`\scriptstyle`**' e '**`\scriptscriptstyle`**'. La prima rappresenta la dimensione più grande, mentre l'ultima genera il carattere più piccolo.

Anche se in condizioni normali non è necessario il controllo diretto della dimensione del testo matematico, è bene conoscere questa possibilità che può rivelarsi utile in presenza di formule particolarmente complesse in cui alcune parti rischiano di diventare troppo piccole per la lettura.

A titolo di esempio viene mostrata una frazione piuttosto articolata, in due modi differenti, anche se le frazioni vengono descritte in un'altra sezione:

```
$$x+{y\over x+{y\over x+{y\over x+{y\over x+{y\over
z}}}}}$$\par
$$x+{y\over \displaystyle x+{y\over \displaystyle x+{y\over
\displaystyle x+{y\over \displaystyle x+{y\over z}}}}}$$\par
```

$$x+\cfrac{y}{x+\cfrac{y}{x+\cfrac{y}{x+\cfrac{y}{x+\cfrac{y}{z}}}}}$$

$$x+\cfrac{y}{x+\cfrac{y}{x+\cfrac{y}{x+\cfrac{y}{x+\cfrac{y}{z}}}}}$$

Esiste la possibilità di usare '**\strut**', anche nell'ambito matematico, ma questo non previene la riduzione della dimensione dei caratteri. Viene mostrato lo stesso esempio in cui si controlla lo spazio verticale disponibile con l'aiuto di '**\strut**':

```
\hsize=14cm
$$x+{y\over \strut x+{y\over \strut x+{y\over \strut
x+{y\over \strut x+{y\over z}}}}}$$\par
```

$$x+\cfrac{y}{x+\cfrac{y}{x+\cfrac{y}{x+\cfrac{y}{x+\cfrac{y}{z}}}}}$$

49.17.3 Punteggiatura

Dal momento che nell'ambiente matematico non si tiene conto degli spazi inseriti nel testo, lo spazio attorno ai segni di punteggiatura è gestito automaticamente. Si può osservare l'esempio seguente:

```
$123456.7890$\par
$123456,7890$\par
$123456:7890$\par
$123456;7890$\par
$123456{,}7890$\par
```

Come si può intuire, se si vuole usare la virgola come separatore della parte intera da quella rimanente, occorre delimitarla in un gruppo a sé stante, per evitare che gli sia aggiunto uno spazio superfluo alla sua destra.

$$123456.7890$$
$$123456,7890$$
$$123456:7890$$
$$123456;7890$$
$$123456{,}7890$$

Un discorso particolare va fatto per i due punti, che possono servire per indicare una divisione, oppure un assegnamento. Si osservi l'esempio seguente in cui si riprende un pezzo di quanto già mostrato:

```
$123456:7890$\par
$123456:=7890$\par
$123456\colon 7890$\par
```

In pratica, per ottenere i due punti «normali», a cui segue un po' di spazio, si deve usare l'istruzione '**\colon**'.

$$123456:7890$$
$$123456:=7890$$
$$123456\colon 7890$$

49.17.4 Frazioni e simili

Le frazioni si indicano solitamente in uno dei due modi seguenti:

numeratore* / *denominatore

numeratore `\over` ***denominatore***

Il secondo modo è quello più gradevole esteticamente. È importante osservare che l'istruzione '**\over**' interviene su tutto il gruppo in cui è contenuta. Si osservino gli esempi seguenti:

- `$${x+y \over z}$$`

$$\frac{x+y}{z}$$

- `$${x \over y+z}$$`

$$\frac{x}{y+z}$$

- `$${a \over x} + {b \over x} = {a+b \over x}$$`

$$\frac{a}{x}+\frac{b}{x}=\frac{a+b}{x}$$

Oltre alle frazioni vere e proprie, ci sono altre situazioni simili:

x `\atop` ***y***

x `\choose` ***y***

x `\above` ***spessore y***

Nel primo caso, si ottiene la sovrapposizione di ***x*** sopra ***y***, senza la linea orizzontale tipica di una frazione; nel secondo si ottiene la stessa cosa, ma il tutto viene anche racchiuso tra parentesi; nell'ultimo caso, si ottiene una cosa simile alla frazione, dove si specifica lo spessore della linea di separazione. Seguono alcuni esempi:

- `$${x+y \atop z}$$`

$$\begin{matrix}x+y\\z\end{matrix}$$

- `$${x+y \choose z}$$`

$$\binom{x+y}{z}$$

- `$${x+y \above 1mm z}$$`

$$\begin{matrix}x+y\\ \hline z\end{matrix}$$

Si osservi che anche in questo caso, le istruzioni mostrate spezzano il gruppo in cui sono inserite, pertanto è necessario racchiudere l'espressione tra parentesi graffe.

La dimensione del testo che si trova a essere spezzata con queste istruzioni viene ridotta automaticamente. Se si vuole evitare questo comportamento, si possono usare le istruzioni già descritte per il controllo esplicito della dimensione.

49.17.5 Apici e pedici

I pedici e gli apici si ottengono con i simboli '_' e '**^**' rispettivamente, che intervengono sul carattere oppure sul raggruppamento successivo:

x*_*pedice

x*^*apice

Si osservino gli esempi successivi, in particolare per quanto riguarda la necessità o meno di raggruppare ciò che va messo ad apice o a pedice:

- `$$10^2$$`

$$10^2$$

- `$$10^{20}$$`

$$10^{20}$$

- `$$10^{-2}$$`

$$10^{-2}$$

- `$$x^a_b$$`

$$x^a_b$$

- `$$x^{y^z}$$`

$$x^{y^z}$$

- `$\sum_{x=0}^n x^3$`

$\sum_{x=0}^n x^3$

- `$$\sum_{x=0}^n x^3$$`

$$\sum_{x=0}^n x^3$$

- `$$\int_0^n f(x)$$`

$$\int_0^n f(x)$$

- `$\lim_{n\to 0} {1 \over n} = \infty$`

$\lim_{n\to 0} \frac{1}{n} = \infty$

- `$$\lim_{n\to 0} {1 \over n} = \infty$$`

$$\lim_{n\to 0} \frac{1}{n} = \infty$$

49.17.6 Radici

Le radici si possono ottenere attraverso due modi differenti, a seconda che si tratti delle radici quadrate comuni, oppure di radici di altro genere:

```
\sqrt{contenuto}
```

```
\root n \of {contenuto}
```

Come si intuisce, nel primo caso si ottiene una radice quadrata pura e semplice, mentre nel secondo si dichiara il tipo di radice e successivamente il contenuto. Seguono alcuni esempi:

- `$$\sqrt{x^2+y^2}$$`

$$\sqrt{x^2+y^2}$$

- `$$\root 3 \of {8}$$`

$$\root 3 \of {8}$$

- `$$\sqrt{x+y \over z}$$`

$$\sqrt{x+y \over z}$$

- `$$x^{\sqrt{y}}$$`

$$x^{\sqrt{y}}$$

49.17.7 Sottolineature e sopralineature

Per ottenere delle linee orizzontali sopra o sotto un gruppo di simboli, si usano rispettivamente '**`\overline`**' e '**`\underline`**':

```
\overline{contenuto_posto_sotto_la_linea}
```



```
\underline{contenuto_posto_sopra_la_linea}
```

Seguono alcuni esempi:

- `$$\overline{x+y}$$`

$$\overline{x+y}$$

- `$$\underline{\overline{x+y}}$$`

$$\underline{\overline{x+y}}$$

49.17.8 Funzioni

Quando si vuole fare riferimento al nome standard di una certa funzione matematica, è necessario fare in modo che questo nome appaia con un carattere diverso dal corsivo matematico, per evitare che si possa confondere con il prodotto di una serie di variabili. Per questa ragione e anche per facilitare la lettura del sorgente TeX, esistono delle macro specifiche, con il solo scopo di scrivere il nome della funzione corrispondente usando un carattere tondo normale. La tabella 49.318 elenca brevemente queste macro. Se per qualche ragione si preferiscono dei nomi differenti, si può sempre usare la tecnica del testo nella scatola orizzontale, attraverso l'istruzione '**`\hbox{`*testo*`}`**'. Viene mostrato solo qualche esempio di utilizzo:

- `$$\sin(2x)$$`

$$\sin(2x)$$

- `$$\cos x$$`

$$\cos x$$

- `$$\hbox{sen}(2x)$$`

$$\hbox{sen}(2x)$$

Benché si possa scrivere il nome di una funzione usando la macro '**`\hbox{`*testo*`}`**', TeX offre l'istruzione primitiva '**`\mathop`**', con cui è possibile definire delle macro aggiuntive da affiancare a quelle standard per la scrittura dei nomi di funzione. Si usa normalmente così:

```
\def\nome{\mathop{\rm nome}}
```

Successivamente, è sufficiente fare riferimento alla macro '**`\`*nome*`**' per ottenere il nome della funzione relativa in un contesto matematico. Segue l'esempio della dichiarazione e dell'uso della funzione seno, con il nome «sen», assieme al risultato che si ottiene:

```
\def\sen{\mathop{\rm sen}}
$$\sen(2x)$$
```

$$\sen(2x)$$

Tabella 49.318. Funzioni standard.

Simbolo	Codice	Annotazioni
arccos	$\arccos$	arco-coseno
arcsin	$\arcsin$	arco-seno
arctan	$\arctan$	arco-tangente
arg	$\arg$	argomento
cos	$\cos$	coseno
cosh	$\cosh$	coseno iperbolico
cot	$\cot$	cotangente
coth	$\coth$	cotangente iperbolica
csc	$\csc$	cosecante
deg	$\deg$	gradi
det	$\det$	determinante
dim	$\dim$	dimensione
exp	$\exp$	esponenziale
gcd	$\gcd$	massimo comune divisore
hom	$\hom$	
inf	$\inf$	
ker	$\ker$	
lg	$\lg$	logaritmo base 2
lim	$\lim$	limite
lim inf	$\liminf$	
lim sup	$\limsup$	
ln	$\ln$	logaritmo naturale
log	$\log$	logaritmo base 10
max	$\max$	massimo
min	$\min$	minimo
Pr	$\Pr$	probabilit
sec	$\sec$	secante
sup	$\sup$	
tan	$\tan$	tangente
tanh	$\tanh$	tangente iperbolica

49.17.9 Delimitatori

In matematica si usano tre tipi di parentesi per delimitare delle sottoespressioni; si tratta notoriamente di parentesi tonde, quadre e graffe. Le stesse parentesi sono disponibili nella modalità matematica di TeX, con la particolarità delle parentesi graffe che devono essere usate attraverso due macro speciali: '**\{**' e '**\}**'. In pratica, questa differenza dipende dall'uso speciale che TeX stesso fa di queste parentesi nella sua sintassi.

In condizioni normali, la grandezza delle parentesi usate non è diversa da quella del carattere standard dell'ambiente matematico. Per ottenere parentesi più grandi si possono usare alcune macroistruzioni come '**\bigl**' e '**\bigr**', le quali devono essere seguite immediatamente da una parentesi o da un altro simbolo di delimitazione, allo scopo di ottenerlo più grande del normale. Si osservi l'esempio seguente:

```
$$\Biggl(\biggl\{\Bigl[\bigl(a+b\bigr)\times
c\Bigr]+d\biggr\}\times e\Biggr)$$
```

$$\left(\left\{\left[\left(a+b\right)\times c\right]+d\right\}\times e\right)$$

In modo più semplice si possono usare le istruzioni '**\left**' e '**\right**', le quali adattano automaticamente le dimensioni delle parentesi o di altri delimitatori, in funzione di ciò che devono contenere. Si osservi l'esempio seguente:

```
$$\left(\left\{\left[\left(a+b\right)\times
c\right]+d\right\}\over e\right)$$
```

$$\left(\frac{\left\{\left[\left(a+b\right)\times c\right]+d\right\}}{e}\right)$$

Si osservi che l'altezza delle parentesi non aumenta in presenza di annidamenti successivi, ma soltanto quando la dimensione del contenuto lo richiede.

Tabella 49.323. Delimitatori in sintesi.

Simbolo	Codice
( ) [ ] { }	$($ $)$ $[$ $]$ $\{$ $\}$
( ) [ ] { }	$\bigl($ $\bigr)$ $\bigl[$ $\bigr]$ $\bigl\{$ $\bigr\}$
( ) [ ] { }	$\Bigl($ $\Bigr)$ $\Bigl[$ $\Bigr]$ $\Bigl\{$ $\Bigr\}$
( ) [ ] { }	$\biggl($ $\biggr)$ $\biggl[$ $\biggr]$ $\biggl\{$ $\biggr\}$
( ) [ ] { }	$\Biggl($ $\Biggr)$ $\Biggl[$ $\Biggr]$ $\Biggl\{$ $\Biggr\}$
(x) $\left(\frac{x}{y}\right)$	$\left(x\right)$ $\left(x\over y\right)$
⌊ ⌋ ⌈ ⌉	$\lfloor$ $\rfloor$ $\lceil$ $\rceil$
⟨ ⟩	$\langle$ $\rangle$
/ \	$/$ $\backslash$
∣ ∥	$\vert$ $\Vert$
↑ ↓	$\uparrow$ $\downarrow$
⇑ ⇓	$\Uparrow$ $\Downarrow$
↕ ⇕	$\updownarrow$ $\Updownarrow$

La tabella 49.323 mostra un elenco sintetico dei delimitatori e dell'uso delle istruzioni che consentono di ingrandirli. Naturalmente, le istruzioni '**\left**' e '**\right**', le macro '**\bigl**', '**\bigr**', fino a '**\Biggl**' e '**\Biggr**', possono essere usate con tutti i tipi di delimitatori matematici disponibili, non solo le parentesi comuni.

49.17.10 Matrici e sistemi di equazioni

Le matrici si rappresentano in modo molto semplice attraverso le macroistruzioni '**\pmatrix**' e '**\matrix**'. La differenza tra le due sta solo nel fatto che nel primo caso si ottiene la delimitazione del contenuto con parentesi tonde di altezza adeguata, mentre nel secondo caso queste devono essere inserite espressamente:

```
\[p]matrix{
elemento_1_1 [& elemento_1_2]... elemento_1_m \cr
[elemento_2_1 [& elemento_2_2]... elemento_2_m \cr]
...
elemento_n_1 [& elemento_n_2]... elemento_n_m \cr
}
```

In pratica, il raggruppamento che segue la macro contiene una serie di elementi organizzati in tabella, separati orizzontalmente dal simbolo '**&**' e conclusi alla fine di ogni riga da '**\cr**'. L'esempio seguente mette a confronto due matrici, realizzate con '**\pmatrix**' e '**\matrix**':

```
$$\pmatrix{
a & b & c \cr
d & e+f & g \cr
h & i & j \cr
}$$

$$\left |
\matrix{
a & b & c \cr
d & e+f & g \cr
h & i & j \cr
}
\right |
$$
```

$$\begin{pmatrix} a & b & c \\ d & e+f & g \\ h & i & j \end{pmatrix}$$

$$\begin{vmatrix} a & b & c \\ d & e+f & g \\ h & i & j \end{vmatrix}$$

Gli elementi della matrice sono spaziati in pratica da uno spazio leggermente elastico che può essere forzato facilmente. Per esempio, volendo allineare le celle a sinistra, o a destra, basta usare nel lato opposto uno spazio un po' più elastico:

```
$$\left |
\matrix{
a & b \hfill & c \cr
d & e+f & g \cr
h & \hfill i & j \cr
}
```

```
\right |
$$
```

$$\left|\matrix{a & b & c \cr d & e+f & g \cr h & i & j}\right|$$

Naturalmente, queste macroistruzioni si possono utilizzare anche per qualcosa di diverso dalle matrici, che però è bene appaia incasellato in forma tabellare:

```
$$|x| =
\bigg \{
\matrix{
x & x \geq 0 \cr
-x & x \leq 0 \cr
}
$$
```

$$|x| = \bigg\{ \matrix{ x & x \geq 0 \cr -x & x \leq 0 \cr }$$

Per rappresentare un insieme di «casi», come nella definizione di una funzione, si può usare la macro '**`\cases`**':

```
\cases{ espressione & condizione \cr
    espressione & condizione \cr
    …
    espressione & condizione \cr }
```

Come si vede dal modello sintattico, si distinguono espressioni e condizioni relative: le espressioni sono inserite in un ambiente matematico, mentre le condizioni si trovano in un ambiente di testo normale.

```
$$f(x) = \cases{
    2x & se $x > 0$ \cr
    {3x \over 2}+x^2 & se $x < 0$ \cr
}$$
```

$$f(x) = \begin{cases} 2x & \text{se } x > 0 \\ \frac{3x}{2} + x^2 & \text{se } x < 0 \end{cases}$$

In modo analogo, si usa la macro '**`\eqalign`**' per allineare delle equazioni; in tal caso però, tutti i dati si trovano nell'ambiente matematico:

```
\eqalign{ espressione & espressione \cr
    espressione & espressione \cr
    …
    espressione & espressione \cr
```

Dato il nome (*equation align*) si intende che l'allineamento debba avere luogo, preferibilmente, in corrispondenza del segno di uguaglianza (o disuguaglianza) usato per abbinare le espressioni. Si osservi l'esempio seguente:

$$\begin{aligned} x+y &= y+x \times 1 \\ (x \times y)^1 &= y \times x \\ \frac{x}{y} &\neq \frac{y}{x} \end{aligned}$$

Infine, un gruppo di espressioni può essere rappresentato su più righe attraverso la macro '**`\displaylines`**', in modo molto simile a '**`\matrix`**':

```
\displaylines{ espressione_1 \cr
    espressione_2 \cr
    espressione_3 \cr
    …
    espressione_n \cr }
```

Viene mostrato un esempio con il risultato che si ottiene:

```
$$\displaylines{ x+y = y+x \cr
    x \times y = y \times x \cr
    {x\over y} \not = {y \over x} \cr }
$$
```


$$x+y = y+x$$
$$x \times y = y \times x$$
$$\frac{x}{y} \neq \frac{y}{x}$$

49.17.11 Dichiarazione di teoremi e corollari

È possibile definire facilmente un teorema o un corollario attraverso la macro '**`\proclaim`**', secondo lo schema seguente:

```
\proclaim titolo. definizione
```

In pratica, si distingue una parte iniziale del testo che segue la macro, fino al primo punto, che viene evidenziato in modo più nero rispetto al resto. Evidentemente, la macroistruzione '**`\proclaim`**' viene usata in un ambiente normale (non matematico) e può contenere ambienti matematici nell'ambito della definizione. Si osservi l'esempio seguente:

```
\proclaim Teorema 1. In teoria, non c'\'e differenza tra
pratica e teoria.\par
\proclaim Teorema 2. In pratica, la teoria \'e diversa dalla
pratica.\par
```

Teorema 1. *In teoria, non c'è differenza tra pratica e teoria.*

Teorema 2. *In pratica, la teoria è diversa dalla pratica.*

È importante tenere presente che la dichiarazione non può essere più lunga di una riga.

49.17.12 Equazioni in evidenza

La gestione di equazioni poste nell'ambiente matematico esterno al testo normale, ovvero quello che viene posto in evidenza al centro della giustezza del testo, è particolare.

In modo simile alle matrici, è possibile allineare le equazioni, ma per questo si usa preferibilmente la macroistruzione '**`\eqalign`**', che prevede l'uso di un solo simbolo '**`&`**' per ogni riga. Si osservi l'esempio:

```
$$\eqalign{
a+b &= c        \cr
a+c &= b+c-1    \cr
b+c &= 5        \cr
}$$
```

$$\begin{aligned} a+b &= c \\ a+c &= b+c-1 \\ b+c &= 5 \end{aligned}$$

Eventualmente, la macroistruzione '**`\equalignno`**', consente di mostrare qualche informazione in più in prossimità del margine destro. Si ottiene ciò individuando una colonna aggiuntiva nella tabella immaginaria, come si può vedere dall'esempio:

```
Testo normale, 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.
$$\eqalignno{
a+b &= c        & (1)   \cr
a+c &= b+c-1    & (2)   \cr
b+c &= 5        \cr
}$$
```

Testo normale, 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.

$$\begin{aligned} a+b &= c && (1) \\ a+c &= b+c-1 && (2) \\ b+c &= 5 \end{aligned}$$

Nello stesso modo, la macroistruzione '**`\lequalignno`**' mostra le annotazioni a sinistra:

```
Testo normale, 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.
$$\leqalignno{
a+b &= c        & (1)   \cr
a+c &= b+c-1    & (2)   \cr
b+c &= 5        \cr
```

```
}$$
```

Testo normale, 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.

(1) $$a+b=c$$

(2) $$a+c=b+c-1$$

$$b+c=5$$

Eventualmente, se c'è la necessità di annotare un'espressione che non ha bisogno di essere rappresentata in forma tabellare, si possono usare le istruzioni '**\eqno**' e '**\leqno**':

```
$$a+b = c       \eqno (1)$$
$$b+c = b+c-1   \leqno (2)$$
```

$$a+b=c \tag{1}$$

(2) $$b+c=b+c-1$$

49.18 Riferimenti

«

- Documentazione interna della distribuzione TeX: '`texmf/doc/*`'
- Documentazione interna Info: *info web2c* e *info latex*
- Marco Pratesi, Marco Latini, Michele Antonecchia *LATEX-PDF-Howto*, 2003, *http://www.telug.it/marco/LaTeX-PDF-HOWTO/LaTeX-PDF-HOWTO.html*
- *General documentation about (La)TeX*, *http://tex.loria.fr/english/general.html*
- Michael Doob, *A gentle introduction to TeX*, *http://www.ctan.org/tex-archive/info/gentle/gentle.pdf*
- Daniel M. Zirin, *TeX made easy, Using TeX with plain macro package*, 1993, Zar Limited, *http://tex.loria.fr/general/all.dvi*
- D. R. Wilkins, *Getting started with plain TeX*, 1993, *http://tex.loria.fr/general/pllong.dvi*
- D. R. Wilkins, *Summary of commonly-used features of plain TeX*, 1993, *http://tex.loria.fr/general/plshort.dvi*
- *TeX cookbook*, 1989, MathPro Press, *http://tex.loria.fr/general/cookbook.dvi*
- Sorgenti della definizione del formato '**tex**', contenuti nella directory '`texmf/tex/plain/base/`' di una distribuzione teTeX
- Sorgenti della definizione del formato '**latex**', contenuti nel file '`texmf/tex/latex/base/latex.ltx`' di una distribuzione teTeX
- Claudio Beccari, *LaTeX, Guida a un sistema di editoria elettronica*, Hoepli, 1991, ISBN 88-203-1931-4

[1] È consentito anche l'uso della virgola per separare la parte intera di un numero. Tuttavia, ci sono situazioni in cui ciò non va bene, per cui è meglio usare sempre solo il punto.

[2] TeX ammette che tra il numero e l'unità di misura ci sia dello spazio. Qui si preferisce attaccare l'unità di misura al numero, per evitare confusione con il testo successivo.

[3] Si distingue tra «parametro formale» e «argomento attuale», per indicare, nel primo caso, ciò che si usa nella dichiarazione di una macroistruzione a rappresentare i dati variabili da fornire, mentre nel secondo caso si fa riferimento ai dati che vengono passati effettivamente, in corrispondenza di un certo parametro della dichiarazione formale.

[4] Si possono creare delle alternative anche a simboli di controllo, ma in generale ciò non è conveniente.

[5] Leggendo la tabella, è il caso di osservare che in condizioni normali, la sequenza di controllo '**\$**' genera il simbolo del dollaro, mentre se il testo è in corsivo, si ottiene il simbolo della sterlina inglese.

[6] Si osservi che in questo caso non si può usare un raggruppamento vuoto al posto dello spazio, perché l'accento risulterebbe spostato rispetto al centro della lettera.

[7] Nella lingua italiana sono sufficienti due tipi di trattini.

[8] Questi spazi verticali aggiuntivi possono essere inseriti solo se ci si trova in modalità verticale, in pratica nello spazio tra i paragrafi e non al loro interno.

LaTeX

50.1 LaTeX: introduzione

LaTeX è un pacchetto di macro per TeX molto comune, per facilitare la realizzazione di documenti tipici. Si fa riferimento implicito a questo pacchetto di macro utilizzando un comando specifico per la composizione, come già descritto nella sezione 49.1.3. Pertanto, di solito si ottiene la composizione di un sorgente TeX scritto usando macro LaTeX attraverso il comando seguente:

```
latex file_sorgente
```

Normalmente, il file sorgente scritto con macro LaTeX ha la solita estensione '`.tex`'; inoltre si usano le stesse procedure per la rielaborazione dei file DVI generati dalla composizione.

In generale, la scrittura di un sorgente LaTeX deve seguire le regole imposte dal suo pacchetto di macro. Ciò significa che le istruzioni o le macro del linguaggio TeX originale potrebbero creare dei conflitti, se l'uso di queste non è previsto espressamente.
Quando si usano solo le macro LaTeX, lasciando da parte TeX, queste macro possono essere viste come ***comandi*** del linguaggio LaTeX, mentre le ***macro*** diventano quelle definizioni create con il linguaggio LaTeX stesso.

50.1.1 Comandi e modelli sintattici

La necessità di avere il controllo della situazione, fa sì che LaTeX definisca anche un modo proprio per la creazione di altre macro; inoltre, esiste la possibilità di avere degli argomenti opzionali, che si indicano tra parentesi quadre. In generale, è difficile definire in modo completo come possono essere conformate le chiamate ai comandi e alle macro di LaTeX, comunque il modello sintattico seguente dovrebbe darne un'idea sufficiente per cominciare.

```
\comando[opzioni]...{argomenti}...
```

È già noto il fatto che le parentesi graffe fanno parte dell'istruzione; inoltre, le parentesi quadre devono essere usate per gli argomenti opzionali, ma se queste opzioni mancano, si omettono anche le parentesi quadre stesse. Per esempio, si può scrivere:

```
\documentclass[a4paper]{book}
```

oppure soltanto:

```
\documentclass{book}
```

Infine, c'è anche da considerare che alcuni comandi prevedono l'uso opzionale di un asterisco alla fine del loro nome, come nel caso della dichiarazione del titolo di una sezione:

```
\section{Bla bla bla}
```

Oppure:

```
\section*{Bla bla bla}
```

Il significato dell'asterisco dipende dal comando.

Ricapitolando, un modello sintattico riferito a comandi o macro di LaTeX deve essere letto in modo speciale: le parentesi graffe, le parentesi quadre e l'asterisco fanno parte dell'istruzione; inoltre, se non si indicano opzioni, le parentesi quadre che le delimitano non vanno usate.

50.1.2 Opzioni e argomenti

Fino a questo punto si è visto che le opzioni sono argomenti facoltativi che, se utilizzati, vanno delimitati attraverso delle parentesi quadre. La loro posizione è stabilita dalla sintassi del comando stesso, anche se di solito dovrebbero trovarsi prima degli argomenti normali.

Un comando potrebbe prevedere l'uso di più opzioni in sequenza, o alternate con gli argomenti. Ma oltre a questo, un'opzione potrebbe essere interpretata in modo da estrapolare più sotto-opzioni, delimitate generalmente attraverso una virgola.

Sugli argomenti c'è poco da aggiungere, tranne ripetere che il loro utilizzo è obbligatorio; inoltre, anche in questo caso, ci possono essere situazioni in cui un argomento è composto da più sotto-argomenti separati da virgole.

Segue la descrizione di alcuni esempi.

- `\documentclass{book}`
  Definisce lo stile generale '**book**' per un documento che inizia.
- `\documentclass[a4paper]{book}`
  Definisce lo stile generale '**book**', specificando l'opzione '**a4paper**'.
- `\documentclass[a4paper,12pt]{book}`
  Definisce lo stile generale '**book**', specificando l'opzione '**a4paper,12pt**', che in pratica si traduce in due sotto-opzioni, distinte in base alla presenza della virgola.
- `\newcommand{\dattilografico}[1]{\texttt{#1}}`
  Questo è un comando un po' difficile da interpretare (serve a creare una macro di LaTeX), comunque si può osservare che appare un'opzione dopo un argomento e prima dell'ultimo argomento. È il caso di precisare che in questo momento, '**\dattilografico**' e '**\texttt{#1}**' sono solo stringhe che hanno un qualche valore per il comando '**\newcommand**'.
- `\epsfig{file=prova,height=3cm,angle=0}`
  Questo comando non riguarda direttamente LaTeX, ma proviene da un pacchetto che comunque lo accompagna. Come si può osservare, c'è solo un argomento, scomposto in tre sotto-argomenti separati da virgole.

50.1.3 Esempio iniziale

Prima di affrontare l'analisi della struttura del sorgente LaTeX, conviene vedere un esempio introduttivo abbastanza completo:

```
\documentclass{article}

% Inizia il preambolo.

\setlength{\textwidth}{11cm}
\setlength{\textheight}{11cm}

% Fine del preambolo.

\begin{document}

% Inizia il documento vero e proprio.

\section{Introduzione a \LaTeX}

Questo \'e un esempio di documento scritto con \LaTeX .
Come si pu\'o vedere \'e gi\'a stato definito uno stile
generale del documento: article.

\subsection{Suddivisione del documento}

Lo stile article prevede una suddivisione in sezioni
sottosezioni ed eventuali sotto-sottosezioni.

\subsection{Paragrafi}

Il testo di un paragrafo termina quando nel sorgente viene
incontrata una riga vuota (una riga bianca).

Questo \'e l'inizio di un nuovo paragrafo e si nota perch\'e
la prima riga \'e leggermente rientrata.

\subsection{Gli ambienti}
```

```
\LaTeX utilizza gli ambienti per definire dei comportamenti
circoscritti a zone particolari del testo.
Per esempio, la centratura si ottiene utilizzando l'ambiente
center.

\begin{center}
Questo \'e un esempio di testo centrato.
\end{center}

% Fine del documento.

\end{document}
```

Supponendo di attribuire a questo file il nome '`primo.tex`', si può procedere con la composizione nel modo seguente:

```
$ latex primo [Invio]
```

Se non vengono rilevati errori, durante l'elaborazione si vedono diverse informazioni sul procedimento della composizione, come già descritto nella sezione 49.1.3 a proposito di TeX. Nello stesso modo di TeX vengono generati dei file contenenti informazioni e la composizione nel file '`primo.dvi`'. Ma a differenza di TeX puro e semplice, si ottiene anche il file '`primo.aux`', contenente delle porzioni di codice da riutilizzare nelle composizioni successive.

Il risultato della composizione di questo esempio si può vedere nella figura 50.11. La conversione del file DVI in qualcosa di più conveniente, è descritta nella sezione 26.3.

Figura 50.11. Il risultato della composizione del sorgente LaTeX di esempio.

1 Introduzione a LaTeX

Questo è un esempio di documento scritto con LaTeX. Come si può vedere è già stato definito uno stile generale del documento: article.

1.1 Suddivisione del documento

Lo stile article prevede una suddivisione in sezioni sottosezioni ed eventuali sotto-sottosezioni.

1.2 Paragrafi

Il testo di un paragrafo termina quando nel sorgente viene incontrata una riga vuota (una riga bianca).

Questo è l'inizio di un nuovo paragrafo e si nota perché la prima riga è leggermente rientrata.

1.3 Gli ambienti

LaTeXutilizza gli ambienti per definire dei comportamenti circoscritti a zone particolari del testo. Per esempio, la centratura si ottiene utilizzando l'ambiente center.

Questo è un esempio di testo centrato.

50.1.4 Sopravvivere nel caos

Nella sezione 49.1 è descritto in che modo si comporta TeX quando si presentano degli errori di composizione. Il pacchetto di macro che compone LaTeX aggiunge informazioni in fase di composizione. Per esempio, si può provare a realizzare un file contenente dei problemi, per vedere come si comporta LaTeX effettivamente:

```
\documentstyle{article}
\begin{document}
\section{Problemi con \LaTeX}
Quanti problemi con \LaTeX!
\end{document}
```

Leggendo il file delle registrazioni (il file con estensione '`.log`') si può trovare un avvertimento che riguarda il comando '**`\documentstyle`**', considerato superato e da non utilizzare.

```
         Entering LaTeX 2.09 COMPATIBILITY MODE
*****************************************************************
```



```
   !!WARNING!!    !!WARNING!!    !!WARNING!!    !!WARNING!!

This mode attempts to provide an emulation of the LaTeX 2.09
author environment so that OLD documents can be successfully
processed. It should NOT be used for NEW documents!

New documents should use Standard LaTeX conventions and start
with the \documentclass command.

Compatibility mode is UNLIKELY TO WORK with LaTeX 2.09 style
files that change any internal macros, especially not with
those that change the FONT SELECTION or OUTPUT ROUTINES.

Therefore such style files MUST BE UPDATED to use
        Current Standard LaTeX: LaTeX2e.
If you suspect that you may be using such a style file, which
is probably very, very old by now, then you should attempt to
get it updated by sending a copy of this error message to the
author of that file.
*****************************************************************
```

Un avvertimento non è niente di eccezionalmente grave, soprattutto se poi non pregiudica la riuscita della composizione. Ma un avvertimento può segnalare il sorgere di un problema che più avanti può aggravarsi e diventare insuperabile. Se nell'esempio mostrato sopra si aggiunge un comando incompatibile si arriva al punto di crisi.

```
\documentstyle{article}
\usepackage{epsfig}
\begin{document}
\section{Problemi con \LaTeX}
Quanti problemi con \LaTeX!
\end{document}
```

Come si vede, è stato aggiunto il comando '**`\usepackage{epsfig}`**', il cui scopo è solo quello di incorporare lo stile '`epsfig.sty`' che si trova da qualche parte, dove LaTeX può trovarlo.

Ciò che dovrebbe succedere è che lo stile richiesto sia incompatibile con una vecchia versione di LaTeX, oppure che sia incompatibile con il funzionamento che si impone a LaTeX quando si utilizza il comando '**`\documentstyle`**'.

```
! LaTeX Error: LaTeX2e command \usepackage in LaTeX 2.09 document.

See the LaTeX manual or LaTeX Companion for explanation.
Type  H <return>  for immediate help.
```

Ecco la segnalazione di errore. Fortunatamente, viene indicato precisamente il punto in cui appare l'errore, esattamente come succede con la composizione in TeX puro e semplice: si tratta proprio del comando '**`\usepackage`**' che appare nella riga numero due del testo sorgente.

```
l.2 \usepackage
                {epsfig}
```

Subito dopo appare l'invito consueto di TeX, che chiede all'utente di prendere una decisione al riguardo. Per il resto, tutto funziona come per TeX.

A differenza dell'utilizzo di TeX puro e semplice, può succedere che LaTeX mostri degli errori nell'ambito delle sue stesse macro, quando ci si spinge ai limiti delle possibilità di queste. Naturalmente, in questi casi occorre cercare il problema nelle istruzioni usate nel file sorgente, lasciando stare i file che compongono il pacchetto di macro di LaTeX.

50.1.5 Struttura essenziale di un documento LaTeX

Il sorgente di un documento scritto in TeX, utilizzando il pacchetto di macro LaTeX, ha una struttura che segue delle regole precise. La prima cosa a essere definita è il tipo di documento, ovvero lo stile generale a cui si vuole fare riferimento. A questo segue eventualmente un preambolo, cioè l'indicazione più o meno facoltativa di altri ele-

menti stilistici insieme alle informazioni che servono a comporre il titolo del documento. Quindi inizia il documento vero e proprio.

Lo stile generale del documento viene definito all'inizio del sorgente LaTeX attraverso la dichiarazione seguente:

```
\documentclass[opzioni]{classe}
```

Per esempio:

```
\documentclass[a4paper,11pt]{book}
```

definisce l'uso della classe '**book**', utilizzando un foglio A4 con il corpo normale dei caratteri a 11 punti tipografici; mentre il comando

```
\documentclass{book}
```

definisce l'uso della classe '**book**', senza opzioni, che così viene impostata in modo predefinito.

Il preambolo è quella parte di sorgente LaTeX che sta tra la dichiarazione della classe (o dello stile generale) e la dichiarazione di inizio del documento. Normalmente viene usata per specificare l'utilizzo di stili aggiuntivi e per l'inserimento di tutti quegli elementi che compongono il titolo del documento e gli indici eventuali.

Una dichiarazione molto importante del preambolo è l'inclusione di uno stile aggiuntivo, secondo la sintassi seguente:

```
\usepackage[opzioni]{pacchetto}
```

Le opzioni utilizzabili dipendono dal tipo particolare di stile a cui si fa riferimento. Un file di stile può anche essere scritto dall'utilizzatore, solitamente partendo da un altro già esistente.

```
\usepackage[utf8x]{inputenc}
\usepackage[T1]{fontenc}
```

L'esempio mostra l'inclusione del pacchetto '**inputenc**' allo scopo di ammettere la codifica dei caratteri UTF-8 nel sorgente LaTeX, assieme al pacchetto '**fontenc**' per ottenere una composizione con un tipo di carattere che contenga le lettere accentate e i simboli speciali più importanti utilizzati in Europa.

Così come è possibile aggiungere altri stili, è possibile utilizzare direttamente delle dichiarazioni riferite a singoli elementi stilistici. Nell'esempio presentato all'inizio si utilizzavano due dichiarazioni:

```
\setlength{\textwidth}{11cm}
\setlength{\textheight}{11cm}
```

In questo caso, si definiva la larghezza e l'altezza del testo, senza fare riferimento a un formato standard.

Il preambolo serve anche per definire gli elementi che fanno parte del titolo del documento. Questi dipendono dal tipo di stile generale utilizzato, ma di solito comprendono almeno il titolo, l'autore e la data, come nell'esempio seguente:

```
\title{Usare \LaTeX}
\author{Pinco Pallino}
\date{11/11/2011}
```

L'inizio del documento è contrassegnato dalla dichiarazione '**\begin{document}**' e la fine da '**\end{document}**'. Tutto quello che appare dopo la conclusione del documento viene semplicemente ignorato.

Subito dopo l'apertura del documento viene collocata normalmente l'istruzione di creazione del titolo, '**\maketitle**', seguita eventualmente da quello di creazione dell'indice, '**\tableofcontents**'.

```
\begin{document}
\maketitle
\tableofcontents
…
…
\end{document}
```

Il corpo del documento può essere normalmente suddiviso, a seconda del tipo di classe utilizzato.


```
\livello_di_suddivisione[opzioni_eventuali]{titolo_della_suddivisione}
```

I nomi dei livelli di suddivisione possono essere i seguenti, elencati in ordine decrescente di importanza:

- '**part**'
- '**chapter**'
  (solo per le classi '**book**' e '**report**')
- '**section**'
- '**subsection**'
- '**subsubsection**'
- '**paragraph**'
- '**subparagraph**'

Per esempio,

```
\section{Introduzione a \LaTeX}
```

definisce l'inizio di una sezione che ha il titolo indicato tra le parentesi graffe.[1]

In particolare esiste un comando speciale, '**\appendix**' che viene utilizzato così, senza opzioni, esclusivamente per modificare il modo in cui vengono numerate le suddivisioni, che da quel punto vengono trattate come parte di un'appendice.

```
\chapter{Bla bla bla}
…
\appendix
\chapter{Appendice bla bla}
…
```

L'esempio mostra proprio questo: il capitolo denominato «Appendice bla bla» è la prima appendice.

50.1.6 Composizione per approssimazione successiva

Tra i vari automatismi offerti da LaTeX, esiste la possibilità di generare degli indici e dei riferimenti incrociati. Per ottenere questo risultato, LaTeX deve annotare inizialmente la posizione dei riferimenti da leggere poi in una fase successiva.

Queste annotazioni vengono fatte in file ausiliari, con estensione '`.aux`' e altre, che vengono letti automaticamente attraverso la ripetizione del processo di composizione. Osservando il file ausiliario principale generato dalla composizione dell'esempio iniziale, si può notare che si tratta di istruzioni LaTeX:

```
\relax
\@writefile{toc}{\contentsline {section}{\numberline {1}Introduzione a \LaTeX  }{1}}
\@writefile{toc}{\contentsline {subsection}{\numberline {1.1}Suddivisione del documento}{1}}
\@writefile{toc}{\contentsline {subsection}{\numberline {1.2}Paragrafi}{1}}
\@writefile{toc}{\contentsline {subsection}{\numberline {1.3}Gli ambienti}{1}}
```

Nel caso particolare dell'esempio mostrato, si tratta solo di istruzioni necessarie alla costruzione dell'indice generale; intuitivamente si comprende il significato dei dati che appaiono: il numero della sezione, il titolo e la pagina in cui si trova.

Gli indici e i riferimenti incrociati vanno a modificare l'impaginazione; si pensi al caso dell'indice generale, dapprima assente del tutto, che in una seconda fase può impegnare da solo più pagine. Pertanto, sono indispensabili tre rielaborazioni per arrivare a una composizione ragionevolmente corretta: la prima fase genera i dati che inizialmente sono assenti del tutto; la seconda incorpora i dati, che però si riferiscono all'impaginazione precedente, in cui mancavano, rigenerando una nuova serie di dati aggiornati; la terza rielaborazione utilizza i dati aggiornati nella seconda fase. Teoricamente, una quarta elaborazione potrebbe generare un risultato ancora più sicuro, che però di solito non risulta necessario.

È importante chiarire che LaTeX non dispone di un procedimento automatico in grado di avviare e riavviare la composizione per tutte le volte in cui ciò possa risultare necessario, pertanto è l'utente stesso che riavvia la composizione in base alle esigenze:

```
$ latex file_sorgente [Invio]
$ latex file_sorgente [Invio]
$ latex file_sorgente [Invio]
```

50.2 Personalizzazione

Come accennato all'inizio del capitolo, LaTeX mette a disposizione dei comandi appositi per la creazione di nuove macro. In queste sezioni viene mostrato come creare e ridefinire una macro di LaTeX e come importare un file di stile, ovvero un file contenente delle macro.

In particolare, negli esempi si fa riferimento spesso a comandi comuni di LaTeX che però non sono ancora stati descritti. In questa fase è sufficiente comprendere i meccanismi con cui si definiscono le macro e si utilizzano i file esterni.

50.2.1 Definizione e ridefinizione di macro

Nel momento in cui si lavora con documenti di grandi dimensioni, oppure si sta preparando una ***veste grafica*** per le proprie pubblicazioni, è importante creare una serie di istruzioni personalizzate per la creazione di ambienti, anche se queste non sono altro che una copia di istruzioni già esistenti. Il vantaggio di questo modo di procedere sta nella possibilità successiva di cambiare tutta la veste grafica semplicemente modificando il funzionamento delle istruzioni personalizzate.

LaTeX prevede anche la possibilità di ridefinire istruzioni già esistenti, ma in tal caso è importante attribuire un senso particolare (e personale) a quelle istruzioni.

L'istruzione '**\newcommand**' permette di creare un comando nuovo, mentre '**\renewcommand**' permette di ridefinirne uno già dichiarato in precedenza.

```
\newcommand{comando}[n_argomenti]{definizione}
```

```
\newcommand{comando}[n_argomenti][primo_argomento_fisso]{definizione}
```

```
\renewcommand{comando}[n_argomenti]{definizione}
```

```
\renewcommand{comando}[n_argomenti][primo_argomento_fisso]{definizione}
```

Se il comando può avere degli argomenti, è necessario indicarne la quantità, attraverso una sola cifra numerica da uno a nove; inoltre, è possibile specificare un contenuto fisso del primo argomento, che viene aggiunto al primo argomento della chiamata. Quando si prevedono degli argomenti, questi vengono inseriti nella stringa di definizione del comando utilizzando delle metavariabili nella forma '**#*n***', dove il numero rappresenta l'***n***-esimo argomento.

In generale, non c'è alcun motivo di usare la dichiarazione in cui si prevede un primo argomento fisso, dal momento che quel testo può essere inserito direttamente nella definizione del comando.

Una volta dichiarato o ridichiarato il comando, questo può essere utilizzato attraverso una sintassi riassumibile nel modo seguente, dove gli argomenti eventuali, appaiono tra parentesi graffe.


```
\comando | \comando{argomento}...
```

L'utilizzo più semplice, riguarda la definizione di un comando senza argomenti, come nell'esempio seguente, dove viene dichiarato '**\bftt**' il cui scopo è quello di iniziare l'uso di un carattere neretto e dattilografico.

```
\newcommand{\bftt}{\bfseries\ttfamily}
```

Come si vede, lo scopo è solo quello di sostituire a '**\bftt**', i comandi '**\bfseries**' e '**\ttfamily**'. Quello che segue è un esempio di come potrebbe essere utilizzato.

```
Il file \bftt mio-file \normalfont contiene...
```

Secondo l'esempio, la parola «mio-file» viene evidenziata in neretto e dattilografico, mentre subito dopo, attraverso il comando '**\normalfont**', riprende lo stile normale.

L'esempio seguente definisce un comando che richiede un argomento. Ciò che si ottiene serve a delimitare una zona in neretto dattilografico.

```
\newcommand{\bftt}[1]{\textbf{\texttt{#1}}}
```

Lo scopo è quello di sostituire al comando '**\bftt{…}**', i comandi '**\textbf{\texttt{…}}**'. Quello che segue è un esempio di come potrebbe essere utilizzato; si può osservare che in questo caso non occorre riconvertire il testo dopo la zona delimitata con le parentesi graffe.

```
Il file \bftt{mio-file} contiene...
```

L'esempio seguente, utilizzando sempre un solo argomento, ha lo scopo di replicarlo cambiandone leggermente lo stile.

```
\newcommand{\triplo}[1]{\textit{#1} \textbf{#1} \texttt{#1}}
```

Utilizzandolo nel modo seguente, si ottiene la ripetizione della parola «tanto» per tre volte, ognuna con uno stile differente: corsivo, neretto e dattilografico.

```
Ti amo \triplo{tanto}...
```

Prima di proseguire, vale la pena di vedere un esempio in cui si dichiara un comando che prevede l'uso di più argomenti.

```
\newcommand{\somma}[3]{\texttt{#1}+\texttt{#2}=\texttt{#3}}
```

Utilizzando il comando appena creato nel modo seguente, si ottiene esattamente il testo '**5+6=11**', dove i numeri sono in dattilografico e i segni sono composti con caratteri normali.

```
\somma{5}{6}{11}
```

Oltre a questo è possibile definire, o ridefinire, degli ambienti da utilizzare nei comandi '**\begin{…}**' e '**\end{…}**'. Per queste operazioni si utilizzano le istruzioni '**\newenvironment**' e '**\renewenvironment**'.

```
\newenvironment{ambiente}[n_argomenti]{def_iniziale}{def_finale}
```

```
\newenvironment{ambiente}[n_argomenti][argomento_predefinito]{def_iniziale}{def_finale}
```

```
\renewenvironment{ambiente}[n_argomenti]{def_iniziale}{def_finale}
```

Se la dichiarazione del nuovo ambiente può avere degli argomenti, è necessario indicarne il numero, attraverso una sola cifra numerica da uno a nove. Questi possono essere utilizzati **solo** nella definizione di inizio, attraverso le metavariabili '**#*n***'. Come nel caso della dichiarazione di un nuovo comando, è possibile specificare il primo argomento predefinito, da utilizzare nel caso non ne sia fornito alcuno.

Una volta dichiarato o ridefinito l'ambiente, questo può essere utilizzato attraverso una sintassi riassumibile nel modo seguente, dove gli argomenti eventuali, appaiono tra parentesi graffe.

```
\begin{ambiente}
testo_contenuto
...
\end{ambiente}
```

```
\begin{ambiente}{argomento}...
testo_contenuto
...
\end{ambiente}
```

L'utilizzo più semplice riguarda la definizione di un ambiente senza argomenti, come nell'esempio seguente, dove viene dichiarato l'ambiente '**\bftt**' il cui scopo è quello di iniziare l'uso di un carattere neretto e dattilografico.

```
\newenvironment{bftt}{\bfseries\ttfamily}{\normalfont}
```

Come si vede, si intende sostituire '**\begin{bftt}**' con il comando '**\bfseries\ttfamily**' e rimpiazzare '**\end{bftt}**' con il comando '**\normalfont**'.

50.2.2 File di stile

La personalizzazione di istruzioni LaTeX può avvenire all'interno del documento stesso, ma generalmente è preferibile creare un file di stile da includere con l'istruzione '**\usepackage{*file_di_stile*}**'.

Quando si crea un nuovo stile conviene fare una copia di uno di quelli già utilizzati da LaTeX e quindi modificarlo.

Quando si utilizza teTeX, questi file di stile dovrebbero trovarsi nella directory '`texmf/tex/latex/base/`'.

50.3 Localizzazione

I problemi legati alla localizzazione del funzionamento di LaTeX riguardano in particolare i termini generati automaticamente dalla composizione (come *Chapter*, *Index* e simili) e la separazione in sillabe.

50.3.1 Terminologia

A seconda dello stile generale del documento che si scrive, quasi sempre, il risultato finale contiene parole inserite automaticamente da LaTeX. Questi termini sono definiti all'interno del file di stile che identifica la classe del documento.

Per modificare questo comportamento si può utilizzare uno stile aggiuntivo, scelto tra quelli contenuti nella directory '`texmf/tex/generic/babel/`', oppure si può creare uno stile personalizzato in cui si ridefiniscono le istruzioni che dichiarano questi termini.

Lo stile aggiuntivo viene caricato normalmente con il comando seguente, posto nel preambolo del documento.

```
\usepackage[italian]{babel}
```

50.3.2 Sillabazione

La sillabazione è configurata attraverso il file '`texmf/tex/generic/config/language.dat`', il quale a sua volta fa riferimento a file contenuti in '`texmf/tex/generic/hyphen/`'.

Se all'interno di questo file sembra non essere attivata la sillabazione per la lingua italiana, conviene modificarlo attraverso il sistema di configurazione di TeX, descritto nella sezione 49.1.

Se la lingua italiana, o quella che interessa, risulta già attivata, oppure se è stata fatta la procedura per attivarla, si può controllare nel file '`texmf/web2c/latex.log`', soprattutto per determinare il numero corrispondente che gli è stato assegnato. Se si trova una riga simile a quella seguente, significa che la sillabazione in italiano è disponibile.

```
\l@italian=\language3
```


50.3.3 Inconvenienti legati alla sillabazione

Se quello che si scrive è un documento tecnico pieno di termini che non fanno parte della lingua italiana, forse conviene disabilitare la sillabazione per evitare la suddivisione di termini stranieri in modo errato. Per farlo, dovrebbe essere sufficiente prevedere l'uso della sillabazione nulla (*nohyphenation*), attivandola nel documento esattamente come si farebbe per un linguaggio normale.

Per esempio, se nel file '`texmf/web2c/latex.log`' si trova la riga seguente,

```
\l@nohyphenation=\language4
```

molto probabilmente dovrebbe essere sufficiente utilizzare il comando '**\language4**' nel documento. Se però, per qualche ragione questo non dovesse funzionare, si possono sempre usare metodi drastici: configurare il file '`texmf/tex/generic/config/language.dat`' commentando tutte le direttive e annullando in ogni caso il sistema di sillabazione.

Eventualmente, se il problema è solo quello di evitare che qualche termine sia separato in sillabe, lo si può inserire in una scatola (come viene descritto meglio in seguito):

```
\mbox{parola}
```

50.3.4 Spaziatura orizzontale

Secondo una regola della tipografia del passato, ormai condannata generalmente, è necessario aumentare lo spazio che divide la fine di un periodo dall'inizio del successivo. Per qualche ragione si trovano ancora documenti in lingua inglese che seguono questa regola, anche quando si tratta di file di testo.

Purtroppo TeX segue quella filosofia e tende a rendere più grande lo spazio orizzontale che c'è tra un punto finale e la parola successiva se questa ha l'iniziale maiuscola. Oltre a tutto, questo sistema crea delle difficoltà nella scrittura degli acronimi o delle abbreviazioni. Si pone rimedio utilizzando anche con LaTeX il comando '**\frenchspacing**' nel preambolo del documento (si veda anche la sezione 49.12 a questo proposito).

50.3.5 Codifica

LaTeX permette l'uso di diverse codifiche, cioè diverse sequenze di simboli nei tipi di carattere utilizzati. Per la precisione, si distingue tra la codifica usata nel sorgente e il tipo di insieme di caratteri da usare. Nel primo caso, si usa normalmente il pacchetto '**inputenc**', nel modo seguente, all'interno del preambolo, in modo da stabilire l'uso della codifica UTF-8 nel sorgente:

```
\usepackage[utf8x]{inputenc}
```

Per quanto riguarda l'insieme di caratteri, il tipo più vecchio è OT1, definito anche *TeX text*; il più recente e più usato è T1, definito anche *TeX text extended*. L'utilizzo dell'insieme di caratteri T1 è necessario se si vuole scrivere un documento che nel sorgente fa uso delle lettere accentate delle lingue europee occidentali.

La selezione dell'insieme di caratteri TeX avviene attraverso il caricamento del pacchetto '**fontenc**', indicando come opzione la sigla dell'insieme desiderato.

```
\usepackage[T1]{fontenc}
```

L'esempio mostra il caricamento dell'insieme di caratteri T1, che è quello che dovrebbe essere utilizzato nella maggior parte dei casi.

50.3.6 Localizzazione in pratica

La definizione del sistema di sillabazione è sempre necessario, mentre si è accennato poco sopra al problema dei termini da tradurre. Il modo più semplice per risolvere il problema della localizzazione (dopo avere sistemato la sillabazione) è quello di utilizzare le istruzioni seguenti nel preambolo.

```
\documentclass…
…
\usepackage[italian]{babel}
\usepackage[utf8x]{inputenc}
\usepackage[T1]{fontenc}
\frenchspacing
…
\begin{document}
…
```

In questo modo, se è stata definita una sillabazione italiana, questa viene attivata automaticamente; i termini come «capitolo», «pagina»,... sono tradotti in italiano; la codifica del sorgente è UTF-8, avendo richiesto l'insieme di caratteri tipografici T1 (quindi il sorgente può essere scritto utilizzando le lettere accentate senza la necessità di utilizzare codici macro particolari).

Se per qualche ragione si vuole redigere un testo multilingua, è possibile utilizzare il pacchetto '**babel**' con l'indicazione di più linguaggi, come nel modo seguente:

```
\usepackage[italian,english]{babel}
```

Successivamente, per selezionarne uno, basta usare il comando '**\selectlanguage**', con il nome prescelto. Ciò ha effetto fino all'uso di un altro comando '**\selectlanguage**' che cambi tale indicazione.

```
…
\selectlanguage{italian}
…
\selectlanguage{english}
…
```

È importante non confondere '**\selectlanguage**' con '**\languagen**'. Nel primo caso si fa riferimento a un comando del pacchetto '**babel**'; nel secondo si fa riferimento alla sillabazione. Il primo è in grado di condizionare il secondo, ma non viceversa: non esiste un «linguaggio» corrispondente alla sillabazione nulla. Se si intende definire la sillabazione nulla occorre passare per un comando '**\languagen**' conforme.

50.4 Variabili e tipi di dati

Nella sezione 49.3 è descritto l'uso delle variabili nel linguaggio TeX, ma quando si usa LaTeX è bene utilizzare comandi appositi, ignorando la gestione che c'è a basso livello. A ogni modo, rimane la possibilità di usare l'istruzione '**\the**' per trasformare il contenuto di una variabile LaTeX in testo della composizione finale.

50.4.1 Lunghezze

Le variabili adibite a contenere le lunghezze, si creano secondo LaTeX attraverso il comando '**\newlength**':

```
\newlength\nome_lunghezza
```

Successivamente, si assegna un valore alle variabili di questo tipo con il comando '**\setlength**':

```
\setlength{\nome_lunghezza}{lunghezza}
```

Come per TeX, le lunghezze che si assegnano in questo modo a una variabile, possono essere rigide oppure elastiche, secondo la stessa notazione di TeX:

```
lunghezza_richiesta [plus lunghezza_in_estensione] ↩
↪ [minus lunghezza_in_contrazione]
```

La lunghezza che costituisce il secondo argomento del comando '**\setlength**', può essere il comando corrispondente a una lunghezza già definita in precedenza; in questo modo si può salvare un valore da richiamare in un momento successivo, quando si intende variare temporaneamente la dimensione di una lunghezza già esistente. Per esempio:

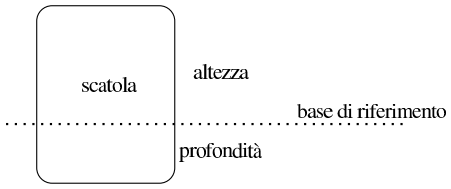
```
\newlength\salvataggio
\setlength{\salvataggio}{\textwidth}
\setlength{\textwidth}{10cm}
…
\setlength{\textwidth}{\salvataggio}
```

Quanto appare sopra, mostra il salvataggio della lunghezza originale di '**\textwidth**' nella variabile '**\salvataggio**', quindi la modifica del valore di '**\textwidth**' temporaneamente, alla fine il ripristino del valore di '**\textwidth**' leggendo il contenuto di '**\salvataggio**'.

L'elenco seguente riepiloga i comandi utili per la creazione e la modifica delle lunghezze:

`\newlength\`*nome_lunghezza*	crea una lunghezza nuova, corrispondente al comando '***nome_lunghezza***';
`\setlength{\`*nome_lunghezza*`}{`*lunghezza*`}`	assegna alla lunghezza indicata il valore dell'ultimo argomento, che può essere sia una lunghezza rigida, sia una lunghezza elastica;
`\addtolength{\`*nome_lunghezza*`}` ↩ ↪ `{`*lunghezza_rigida*`}`	aggiunge alla lunghezza indicata il valore dell'ultimo argomento, che deve essere una lunghezza rigida;
`\settodepth{\`*nome_lunghezza*`}{`*testo*`}`	assegna alla lunghezza il valore della profondità del testo che appare come ultimo argomento;
`\settoheight{\`*nome_lunghezza*`}{`*testo*`}`	assegna alla lunghezza il valore dell'altezza del testo che appare come ultimo argomento;
`\settowidth{\`*nome_lunghezza*`}{`*testo*`}`	assegna alla lunghezza il valore della larghezza del testo che appare come ultimo argomento.

Figura 50.45. Dimensioni di una scatola.



50.4.2 Contatori

LaTeX gestisce dei contatori a cui si accede attraverso parole chiave, nell'ambito di comandi che ne consentono la creazione, la modifica e la lettura. L'elenco seguente riguarda una parte dei comandi coinvolti nella gestione dei contatori:

Comando	Descrizione
`\newcounter{`*nome_contatore*`}` ↩ ↪ `[`*contatore_già_esistente*`]`	crea un contatore con il nome indicato, il cui valore viene azzerato ogni volta che quello opzionale (tra parentesi quadre) viene incrementato;
`\setcounter{`*nome_contatore*`}{`*valore*`}`	assegna il valore indicato al contatore (deve trattarsi di un valore intero, che eventualmente può essere negativo);
`\stepcounter{`*nome_contatore*`}`	incrementa il contatore di una singola unità e azzera eventualmente i contatori che dipendono da questo;

Comando	Descrizione
`\refstepcounter{`*nome_contatore*`}`	si comporta come '**`\stepcounter`**', con la differenza che coinvolge la creazione di un riferimento se seguito dal comando '**`\label`**';
`\addtocounter{`*nome_contatore*`}{`*valore*`}`	aggiunge al contatore il valore indicato (deve trattarsi di un valore intero, che eventualmente può essere negativo);
`\arabic{`*nome_contatore*`}`	traduce il valore del contatore in un numero arabo nella composizione finale;
`\the`*nome_contatore*	quando viene creato un contatore, si crea implicitamente questo comando, con il quale si ottiene il valore del contatore nella composizione finale, espresso in modo predefinito (di solito si tratta di un numero arabo);
`\alph{`*nome_contatore*`}`	traduce il valore del contatore in una lettera minuscola singola, pertanto si possono rappresentare solo valori da 1 a 26;
`\Alph{`*nome_contatore*`}`	traduce il valore del contatore in una lettera maiuscola singola, pertanto si possono rappresentare solo valori da 1 a 26;
`\roman{`*nome_contatore*`}`	traduce il valore del contatore in un numero romano con lettere minuscole, pertanto non si possono rappresentare valori negativi;
`\Roman{`*nome_contatore*`}`	traduce il valore del contatore in un numero romano con lettere maiuscole, pertanto non si possono rappresentare valori negativi;
`\fnsymbol{`*nome_contatore*`}`	traduce il valore del contatore in un simbolo, ma sono disponibili solo nove simboli, pertanto si rappresentano valori da 1 a 9;
`\value{`*nome_contatore*`}`	ottiene il valore del contatore, da usare all'interno di un'espressione (non riguarda la composizione).

Per esempio, si crea un contatore '**`prova`**' con il comando seguente, assegnandogli implicitamente il valore iniziale zero:

```
\newcounter{prova}
```

Al contatore è possibile assegnare esplicitamente un valore:

```
\setcounter{prova}{-35}
```

In alternativa, si può assegnare al contatore il valore di un contatore già esistente; in questo caso si assegna il valore del contatore della pagina:

```
\setcounter{prova}{\value{page}}
```

Infine, si può inserire il valore del contatore nella composizione finale, per esempio in forma di numero arabo normale:

```
\arabic{prova}
```

Per comprendere la dipendenza che ci può essere tra i contatori, viene proposto un esempio più articolato, seguito dal risultato che si

ottiene con la composizione:

```
\newcounter{uno}
\newcounter{due}[uno]

\stepcounter{uno}

\theuno . \thedue

\stepcounter{due}

\theuno . \thedue

\stepcounter{due}

\theuno . \thedue

\stepcounter{uno}

\theuno . \thedue
```

Come si può notare, in questo caso è stata usata la forma '**`\the`***`contatore`*' per visualizzare il valore del contatore preso in esame:

1. 0
1. 1
1. 2
2. 0

Per quanto riguarda l'uso di '**`\refstepcounter`**', le cose si complicano e richiedono la conoscenza dei comandi '**`\label`**' e '**`\ref`**' (sezione 50.10). Qui viene mostrato un esempio che dovrebbe consentire una comprensione intuitiva della cosa:

```
\newcounter{prova}
\stepcounter{prova}
Il contatore ha raggiunto il valore \arabic{prova}.\par
\stepcounter{prova}
Il contatore ha raggiunto il valore \arabic{prova}.\par
\refstepcounter{prova}\label{qui}
Ciao.\par
\stepcounter{prova}
Il contatore ha raggiunto il valore \arabic{prova}.\par

Quando si mostrava la parola ''ciao'', il contatore aveva il
valore \ref{qui}.
```

Si osservi che in questo caso, avendo a che fare con riferimenti incrociati, la composizione deve essere eseguita almeno due volte:

Il contatore ha raggiunto il valore 1.
Il contatore ha raggiunto il valore 2.
Ciao.
Il contatore ha raggiunto il valore 4.
Quando si mostrava la parola "ciao", il contatore aveva il valore 3.

Il comando '**`\the`***`contatore`*' può essere ridefinito, se si desidera ottenere una forma di visualizzazione standard differente:

```
\newcounter{prova}
\renewcommand{\theprova}{\Roman{prova}}
\stepcounter{prova}
Il contatore ha raggiunto il valore \theprova.\par
\stepcounter{prova}
Il contatore ha raggiunto il valore \theprova.\par
```

Il contatore ha raggiunto il valore I.
Il contatore ha raggiunto il valore II.

LaTeX definisce e gestisce automaticamente un certo numero di contatori, a cui può essere conveniente fare riferimento. La tabella seguente elenca i contatori predefiniti più importanti:

Contatore	Descrizione
`part`	contatore delle parti;
`chapter`	contatore dei capitoli;
`section`	contatore delle sezioni;
`subsection`	contatore delle sottosezioni;
`subsubsection`	contatore delle sotto-sottosezioni;

Contatore	Descrizione
`paragraph`	contatore dei paragrafi;
`subparagraph`	contatore dei sottoparagrafi;
`page`	contatore delle pagine;
`equation`	contatore delle equazioni;
`figure`	contatore delle figure;
`table`	contatore delle tabelle;
`footnote`	contatore delle note a piè di pagina;
`enumi` `enumii` `enumiii` `enumiv`	contatori usati negli elenchi numerati.

50.5 Struttura del documento

LaTeX semplifica la scrittura di un documento, rispetto al linguaggio TeX puro e semplice, ma per questo richiede l'uso di una struttura particolare nel sorgente.

50.5.1 Dichiarazione dello stile generale

Lo stile generale del documento viene definito all'inizio del sorgente LaTeX attraverso la dichiarazione seguente:

```
\documentclass[opzioni]{classe}
```

Le classi possono essere:

Classe	Descrizione
`slides`	rappresenta una sequenza di diapositive (presentazioni).
`letter`	rappresenta una lettera vera e propria, dove non si ammette la separazione del testo in sezioni;
`article`	corrispondente a un documento senza tante pretese, ma ugualmente strutturato, che può ammettere la suddivisione in parti e sezioni (sono esclusi i capitoli);
`report`	corrispondente a un documento strutturato in parti e capitoli, inteso come una relazione;
`book`	permette di realizzare un libro suddiviso in parti e capitoli, ma a differenza della classe '**`report`**', le parti e i capitoli iniziano su pagine dispari (destre);

Il comando '**`\documentclass`**' ammette l'uso di una sola opzione, ma al suo interno possono essere indicate diverse sotto-opzioni, rappresentate da delle parole chiave, separate attraverso una virgola. Ogni classe di documento può gestire il suo gruppo particolare di sotto-opzioni, ma in generale, sono disponibili quelle seguenti che dovrebbero essere valide in ogni circostanza.

Dimensione dei caratteri:

Opzione	Descrizione
`10pt`	il testo normale usa caratteri con un corpo di 10 punti (predefinito);
`11pt`	il testo normale usa caratteri con un corpo di 11 punti;
`12pt`	il testo normale usa caratteri con un corpo di 12 punti.

Dimensione del foglio:

Opzione	Descrizione
`a4paper`	formato A4;
`a5paper`	formato A5;
`b5paper`	formato B5;

Opzione	Descrizione
`letterpaper`	formato Lettera (8,5 in × 11 in);
`legalpaper`	formato Legal (8,5 in × 14 in);
`executivepaper`	formato Executive (7,25 in × 10,5 in);.

Altre caratteristiche:

Opzione	Descrizione
`portrait`	orientamento verticale della pagina (predefinito);
`landscape`	orientamento orizzontale della pagina;
`draft`	bozza, evidenzia con una riga verticale il testo che non rispetta i bordi;
`final`	versione finale, senza segnalazioni sul debordamento del testo (predefinito);
`twocolumn`	divide il corpo in due colonne.

Per esempio:

```
\documentclass[a4paper,11pt]{book}
```

definisce l'uso della classe '**book**', utilizzando un foglio A4 con il corpo normale dei caratteri a 11 punti tipografici; mentre il comando

```
\documentclass{book}
```

definisce l'uso della classe '**book**', senza opzioni, che così viene impostata in modo predefinito.

50.5.2 Preambolo

Il preambolo è quella parte di sorgente LaTeX che sta tra la dichiarazione della classe (o dello stile generale) e la dichiarazione di inizio del documento. In questo spazio si possono inserire dei comandi di LaTeX, ma non del testo da comporre. In particolare si inseriscono qui le dichiarazioni di stili aggiuntivi da incorporare, con il comando '**`\usepackage`**':

```
\usepackage[opzioni]{pacchetto}
```

Un gruppo abbastanza comune di comandi che di norma vengono inseriti nel preambolo, è quello con cui si definiscono i dati necessari a costruire poi la pagina del titolo:

Comando	Descrizione
`\title{`*titolo*`}`	titolo del documento (si può usare la sequenza '`\\`' per ottenere un'interruzione di riga);
`\author{`*autore*`}`	autore o autori del documento (si può usare la sequenza '`\\`' per ottenere un'interruzione di riga);
`\date{`*data*`}`	data del documento;
`\thanks{`*annotazione*`}`	nota a piè di pagina riferita al titolo.

50.5.3 Inizio e fine del documento

L'inizio del documento è contrassegnato dalla dichiarazione '**`\begin{document}`**' e la fine da '**`\end{document}`**'. Tutto quello che appare dopo la conclusione del documento viene semplicemente ignorato.

Subito dopo l'apertura del documento viene collocata normalmente l'istruzione di creazione del titolo, '**`\maketitle`**', seguita eventualmente dal comando di creazione dell'indice, '**`\tableofcontents`**':

```
\begin{document}
\maketitle
\tableofcontents
…
…
\end{document}
```

Segue il riepilogo dei comandi che riguardano in particolare l'inserimento dell'indice generale e di quelli specifici di figure e tabelle:

Comando	Descrizione
`\maketitle`	genera la pagina del titolo, traendo i dati dai comandi '**`\title`**', '**`\author`**', '**`\date`**' e '**`\thanks`**' usati nel preambolo;
`\tableofcontents`	inserisce l'indice generale del documento;
`\listoffigures`	elenca le figure che appaiono nel documento;
`\listoftables`	elenca le tabelle che appaiono nel documento.

A seconda della classe del documento, i comandi elencati nella tabella, che servono per la creazione di un indice, prevedono l'inserimento di un titolo automatico, attraverso un comando di dichiarazione di un capitolo o di una sezione non numerati (con un asterisco, come spiegato nella sezione successiva).

Si osservi che, a differenza dell'indice generale, la creazione di un indice analitico è un processo più complicato, descritto nella sezione 50.10.

50.5.4 Suddivisione del documento

Normalmente, il corpo del documento può essere suddiviso, a seconda del tipo di classe utilizzato. Si usano per questo i comandi elencati nella tabella seguente (anche l'asterisco fa parte dell'istruzione):

Comando	Descrizione
`\part[`*opzioni*`]{`*titolo_della_parte*`}` `\part*[`*opzioni*`]{`*titolo_della_parte*`}`	inizia una parte (solo per le classi '**`article`**', '**`report`**' e '**`book`**');
`\chapter[`*opzioni*`]{`*titolo_del_capitolo*`}` `\chapter*[`*opzioni*`]{`*titolo_del_capitolo*`}`	inizia un capitolo (solo per le classi '**`report`**' e '**`book`**');
`\section[`*opzioni*`]{`*titolo_della_sezione*`}` `\section*[`*opzioni*`]{`*titolo_della_sezione*`}`	inizia una sezione;
`\subsection[`*opzioni*`]`↩ ↪`{`*titolo_della_sottosezione*`}` `\subsection*[`*opzioni*`]`↩ ↪`{`*titolo_della_sottosezione*`}`	inizia una sottosezione;
`\subsubsection[`*opzioni*`]`↩ ↪`{`*titolo_della_sotto_sottosezione*`}` `\subsubsection*[`*opzioni*`]`↩ ↪`{`*titolo_della_sotto_sottosezione*`}`	inizia una sotto-sottosezione;
`\paragraph[`*opzioni*`]{`*titolo_del_paragrafo*`}` `\paragraph*[`*opzioni*`]{`*titolo_del_paragrafo*`}`	inizia un paragrafo;
`\subparagraph[`*opzioni*`]`↩ ↪`{`*titolo_del_sottoparagrafo*`}` `\subparagraph*[`*opzioni*`]`↩ ↪`{`*titolo_del_sottoparagrafo*`}`	inizia un sottoparagrafo;
`\appendix`	definisce l'inizio delle appendici.

In pratica, tutti questi comandi, escluso '**`\appendix`**', dichiarano un titolo, che a seconda del caso appartiene all'inizio di una parte, di un capitolo o di una suddivisione inferiore. Come si vede dai modelli sintattici, questi comandi prevedono la presenza di opzioni, che però in pratica non si usano; inoltre, esistono due versioni del comando, con o senza asterisco.

I comandi da '**`\part`**' a '**`\subsection`**', vengono utilizzati per la costruzione dell'indice generale (che si può poi ottenere con il comando '**`\tableofcontents`**'); inoltre, questi comandi prevedono la numerazione automatica delle voci, a seconda del livello di importanza. Se si usano questi comandi con l'asterisco, viene a mancare la numerazione e l'inserimento nell'indice generale. Le suddivisioni di livello inferiore non prevedono la numerazione e l'inserimento nell'indice, pertanto l'asterisco è praticamente privo di effetto.


```
\documentclass{report}
…
\begin{document}

\chapter{\LaTeX}

Bla bla bla bla bla bla bla...

\section{Suddivisione del documento}

Bla bla bla bla bla bla bla...

\section*{Sezione non numerata}

Questa sezione non è numerata e non ha una voce nell'indice
generale.

\subsection{Sottosezioni}

Questa è una sottosezione normale, bla bla bla...

\subsubsection{Sotto-sottosezione}

Questa è una sotto-sottosezione normale, bla bla bla...

\paragraph{Paragrafo normale}

Questo è un paragrafo normale, bla bla bla...

\subparagraph{Sottoparagrafo normale}

Questo è un sottoparagrafo normale, bla bla bla...

\end{document}
```

L'esempio, in cui non appare il preambolo (nel quale si deve definire l'uso della lingua italiana, la codifica adatta e una classe del documento che consenta la suddivisione in capitoli), si traduce nel risultato visibile nella figura 50.69, dove si può osservare in che modo procede la numerazione delle sezioni e l'aspetto normale delle suddivisioni di importanza minore.

Figura 50.69. Esempio nell'uso dei comandi di suddivisione del documento.

Capitolo 1

LaTeX

Bla bla bla bla bla bla bla...

1.1 Suddivisione del documento

Bla bla bla bla bla bla bla...

Sezione non numerata

Questa sezione non è numerata e non ha una voce nell'indice generale.

1.1.1 Sottosezioni

Questa è una sottosezione normale, bla bla bla...

Sotto-sottosezione

Questa è una sotto-sottosezione normale, bla bla bla...

Paragrafo normale Questo è un paragrafo normale, bla bla bla...

Sottoparagrafo normale Questo è un sottoparagrafo normale, bla bla bla...

Quando si incontra il comando '**\appendix**', la numerazione dei capitoli cambia, diventando alfabetica. Se nell'esempio già mostrato si inserisce il comando '**\appendix**' prima della dichiarazione del capitolo, si ottiene il risultato che vi vede nella figura 50.71

```
\appendix

\chapter{\LaTeX}

Bla bla bla bla bla bla bla...

…
```


Figura 50.71. Appendici.

Appendice A

LaTeX

Bla bla bla bla bla bla bla...

A.1 Suddivisione del documento

Bla bla bla bla bla bla bla...

Sezione non numerata

Questa sezione non è numerata e non ha una voce nell'indice generale.

A.1.1 Sottosezioni

Questa è una sottosezione normale, bla bla bla...

Sotto-sottosezione

Questa è una sotto-sottosezione normale, bla bla bla...

Paragrafo normale Questo è un paragrafo normale, bla bla bla...

Sottoparagrafo normale Questo è un sottoparagrafo normale, bla bla bla...

50.5.5 Ambienti

LaTeX introduce un tipo nuovo di comando che serve a definire degli ***ambienti***. L'istruzione '**\begin{*ambiente*}**' delimita l'inizio di un ambiente le cui caratteristiche sono definite dal nome contenuto tra le parentesi graffe; l'istruzione '**\end{*ambiente*}**' delimita la fine dell'ambiente dichiarato in precedenza. Per esempio, l'ambiente '**document**' definisce la zona in cui appare il corpo del documento.

A titolo di esempio, vengono elencati alcuni ambienti comuni, utilizzati frequentemente per definire le caratteristiche di paragrafi particolari:

Comando	Descrizione
`\begin{center}`*testo*`\end{center}`	centra orizzontalmente il testo;
`\begin{quote}`*testo*`\end{quote}` `\begin{quotation}`*testo*`\end{quotation}`	fa rientrare leggermente il testo a destra e a sinistra;
`\begin{verbatim}`*testo*`\end{verbatim}`	mantiene il testo esattamente come digitato nel sorgente.

L'esempio seguente mostra i comandi necessari a centrare il testo «Ciao a tutti!».

```
\begin{center}
Ciao a tutti!
\end{center}
```

50.6 LaTeX: la pagina

La pagina di LaTeX viene definita automaticamente, in base alle opzioni usate nella definizione della classe del documento; eventualmente è possibile intervenire per cambiare l'impostazione predefinita, ma in tal caso tutto diventa piuttosto complesso.

50.6.1 Dimensioni della pagina

Le dimensioni della pagina sono definite da una serie di lunghezze rigide che possono essere modificate facilmente, ma per questo occorre conoscere le varie competenze di queste variabili. In parti-

colare, occorre tenere presente che, a seconda della classe, ci può essere o meno una distinzione tra pagine destre e sinistre.

Figura 50.74a. Lunghezze che definiscono la forma della pagina: pagina sinistra, o pagina pari.

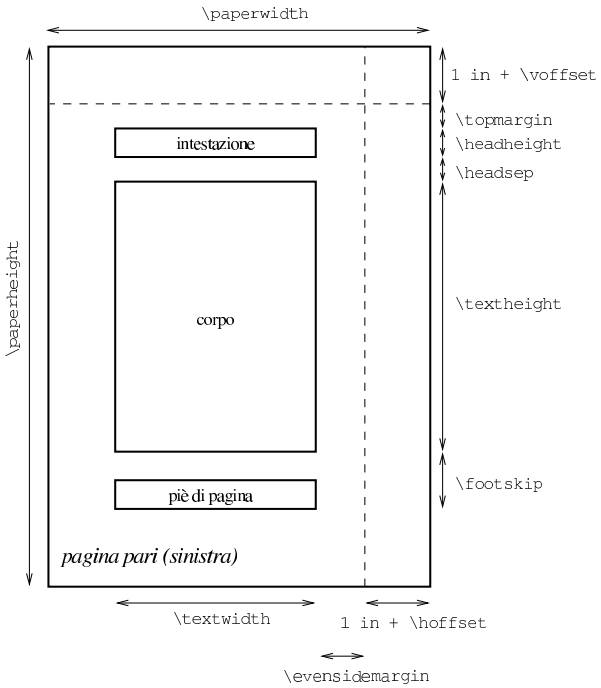
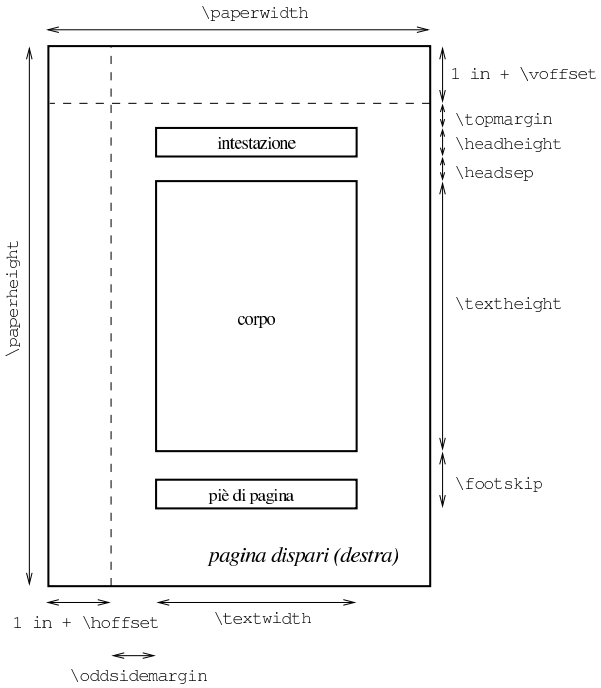


Figura 50.74b. Lunghezze che definiscono la forma della pagina: la pagina destra o pagina dispari.



La figura 50.74 mostra schematicamente la competenza delle varie lunghezze utilizzate; in particolare, si osservi che si fa riferimento sempre a pagine destre, salvo il caso di classi che distinguono tra pagine destre e sinistre. La tabella 50.76 descrive le varie lunghezze, mostrando il valore predefinito in presenza della classe '**book**', con l'opzione '**a4paper**'.

Tabella 50.76. Elenco dei comandi che intervengono nella definizione delle dimensioni della pagina.

Lunghezza	Valore predefinito	Descrizione
`\paperwidth`	614.295pt	Larghezza del foglio di carta su cui si stampa.
`\paperheight`	794.96999pt	Altezza del foglio di carta su cui si stampa.
`\hoffset`	0.0pt	Scostamento orizzontale dal bordo interno, dopo il margine di 2,54 cm (1 in).
`\voffset`	0.0pt	Scostamento verticale dal bordo superiore, dopo il margine di 2,54 cm (1 in).
`\oddsidemargin`	35.0pt	Margine dopo lo scostamento orizzontale, per le pagine destre.
`\evensidemargin`	89.0pt	Margine dopo lo scostamento orizzontale, per le pagine sinistre.
`\textwidth`	398.33858pt	Larghezza del corpo in cui si inserisce il testo.
`\textheight`	550.0pt	Altezza del corpo in cui si inserisce il testo.
`\columnwidth`	398.33858pt	Larghezza della colonna (se la pagina contiene una sola colonna di testo, la larghezza corrisponde a quella del corpo a disposizione).
`\columnsep`	10.0pt	Distanza tra le colonne, quando la composizione è a due colonne.
`\headheight`	12.0pt	Altezza dell'intestazione.
`\topmargin`	22.0pt	Margine dopo lo scostamento verticale, fino alla parte superiore della riga di intestazione.
`\headsep`	18.06749pt	Spazio di separazione verticale tra la base della riga di intestazione e la parte superiore del corpo.
`\footskip`	25.29494pt	Spazio tra la parte inferiore del corpo e la base della riga del piè di pagina.

È importante osservare i margini ('**`\topmargin`**', '**`\oddsidemargin`**' o '**`\evensidemargin`**') fanno riferimento a un punto di inizio spostato di un pollice, sia dal margine interno, sia dall'alto; pertanto, se si vogliono avere margini inferiori a 2,54 cm, i valori delle lunghezze di scostamento relative possono essere negativi.

50.6.2 Flusso verticale del testo, colonne e salti pagina

Con LaTeX, il testo viene distribuito verticalmente in modo analogo a quanto fa già TeX; tuttavia, cambiano i comandi per interrompere le righe e per il salto pagina, inoltre si inseriscono comandi per richiedere di non saltare pagina in un certo punto. La tabella seguente elenca i comandi che consentono un controllo del salto pagina:

Comando	Descrizione
`\newline`	interrompe il flusso del blocco per iniziare una riga nuova;
`\linebreak[0|1|2|3|4]`	richiede un'interruzione di riga con un grado di intensità pari al numero tra parentesi quadre (il numero maggiore richiede l'interruzione di riga con maggiore insistenza);
`\newpage`	salta all'inizio della prossima colonna;
`\clearpage`	esegue un salto pagina, assicurando che vengano inseriti gli oggetti fluttuanti che non hanno ancora trovato posto nella composizione;
`\cleardoublepage`	esegue un salto pagina, in modo da iniziare con una pagina destra (dispari), assicurando che vengano inseriti gli oggetti fluttuanti che non hanno ancora trovato posto nella composizione;
`\pagebreak[0|1|2|3|4]`	richiede un salto pagina con un grado di intensità pari al numero tra parentesi quadre (il numero maggiore richiede il salto pagina con maggiore insistenza);

Comando	Descrizione
`\nopagebreak[0|1|2|3|4]`	richiede che non venga fatto un salto pagina nel punto in cui appare il comando, con un grado di intensità pari al numero tra parentesi quadre (il numero maggiore richiede che non si faccia il salto pagina con maggiore insistenza);
{`\samepage` *testo*}	richiede che il testo indicato rimanga nella stessa pagina;
`\vfill`	inserisce uno spazio verticale elastico.

I comandi '**`\newline`**', '**`\newpage`**', '**`\clearpage`**' e '**`\cleardoublepage`**' sono drastici (il particolare legato agli oggetti fluttuanti viene chiarito nel capitolo in cui si descrive l'inserimento di immagini e di tabelle fluttuanti), mentre '**`\linebreak`**', '**`\nolinebreak`**', '**`\pagebreak`**' e '**`\nopagebreak`**', con o senza opzione, dipendono dalla calcolo di convenienza che viene fatto in fase di composizione.

Per tenere unita una porzione di testo, il metodo più efficace è dato dal comando '**`\samepage`**', che deve essere inserito in un raggruppamento tra parentesi graffe, come si vede nella tabella riepilogativa. Tuttavia, anche in questo caso, non si può garantire che tutto il pezzo richiesto rimanga nella stessa pagina: la probabilità che sia inserita un'interruzione di pagina aumenta con l'allontanarsi dal comando.

Tra questi comandi, '**`\vfill`**' non serve per ottenere o per evitare dei salti pagina, ma consente di spostare in basso del testo, quando viene usato con la richiesta di un salto pagina. Si osservi l'esempio seguente:

```
\noindent Bla bla bla bla bla bla...

\vfill

\noindent Copyright © 2002-2022 Tizio Tizi

\noindent 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.

\noindent This work is distributed in the hope that it will
be useful, but WITHOUT ANY WARRANTY; without even the
implied warranty of MERCHANTABILITY or FITNESS FOR A
PARTICULAR PURPOSE. See the GNU General Public License for
more details.

\noindent You should have received a copy of the GNU General
Public License along with this work; if not, write to the
Free Software Foundation, Inc., 675 Mass Ave, Cambridge, MA
02139, USA.

\newpage
```

La figura 50.79 mostra il risultato della composizione, simulando una pagina molto corta, per questioni di spazio.

Figura 50.79. Allineamento in basso del testo con l'aiuto di '**`\vfill`**'.

Bla bla bla bla bla bla...

È da osservare che, a differenza del linguaggio TeX puro e semplice, i comandi per il salto pagina provvedono da soli ad aggiungere uno spazio elastico adeguato, per evitare che il contenuto della pagina venga allungato in modo inopportuno.

I comandi che richiedono un salto pagina, non si possono cumulare. In pratica, due comandi '**`\newpage`**' in sequenza, generano un salto solo. Per riuscire a ottenere un salto pagina aggiuntivo, occorre scrivere qualcosa, come uno spazio non interrompibile, prima di un nuovo salto pagina. È per questo che esiste il comando '**`\cleardoublepage`**' che permette di raggiungere l'inizio di una pagina destra.

LaTeX consente di controllare in modo complessivo l'allineamento verticale, attraverso due comandi molto semplici:

Comando	Descrizione
`\raggedbottom`	lascia che il testo si distribuisca verticalmente in modo naturale, senza tentare di allungarlo per occupare tutto lo spazio a disposizione;
`\flushbottom`	se possibile, allunga verticalmente gli spazi per occupare tutto lo spazio verticale della pagina.

LaTeX utilizza in modo predefinito una o l'altra forma, a seconda della classe del documento. Come già accennato, i comandi per il salto pagina inseriscono automaticamente uno spazio elastico sufficiente a evitare che in quel punto si tenti di allungare il testo fino alla fine della pagina; tuttavia, ci sono situazioni in cui con la modalità corrispondente al comando '**`\flushbottom`**' si ottengono pagine troppo allungate.

In condizioni normali, LaTeX genera una composizione su una sola colonna verticale; tuttavia è possibile distribuire il testo su due colonne, con l'uso di due comandi particolari, che consentono di cambiare da una modalità all'altra:

Comando	Descrizione
`\onecolumn`	inizia una nuova pagina con una sola colonna;
`\twocolumn`	inizia una nuova pagina con due colonne;
`\twocolumn[`*testo*`]`	inizia una nuova pagina con due colonne, mettendo il testo che appare tra parentesi quadre all'inizio, ma distribuito orizzontalmente su entrambe le colonne.

L'esempio seguente consente di vedere in pratica una composizione a due colonne. Si può osservare il fatto che la frase iniziale viene posta a cavallo delle due colonne.

```
\twocolumn[Bla bla bla bla... questo testo va a cavallo
delle due colonne]

Qui inizia la suddivisione del testo in due colonne
verticali. 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 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...

\newpage

Qui inizia un'altra colonna, a seguito dell'uso del comando
di salto pagina semplice. Bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla...
```

Come si può osservare nella composizione finale che appare nella figura 50.83, il comando '**`\newpage`**' genera un salto fino all'inizio della colonna successiva, indipendentemente dal fatto che questa si trovi nella pagina corrente o in quella successiva. Al contrario, i comandi '**`\clearpage`**' e, necessariamente, '**`cleardoublepage`**', fanno ricominciare la composizione effettivamente in una pagina nuova.

Figura 50.83. Esempio di composizione a due colonne.

Bla bla bla bla... questo testo va a cavallo delle due colonne

Qui inizia la suddivisione del testo in due colonne verticali. 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 bla...

Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla...

Qui inizia un'altra colonna, a seguito dell'uso del comando di salto pagina semplice. Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla...

Il comando '**`\twocolumn`**' crea due colonne di larghezza uguale, utilizzando lo spazio definito dalla lunghezza '**`\textwidth`**', meno lo spazio di '**`\columnsep`**'. Pertanto, se si desidera cambiare la larghezza dello spazio interno alle colonne occorre intervenire prima nella lunghezza '**`\columnsep`**' e quindi usare il comando '**`\twocolumn`**'.

50.6.3 Stile della pagina

LaTeX definisce in modo automatico l'aspetto della riga di intestazione e di quella del fondo pagina, in base alla classe del documento, inserendo eventualmente delle varianti all'inizio delle parti e dei capitoli. I comandi elencati nella tabella seguente, consentono di interferire nello stile della pagina:

Comando	Descrizione
`\pagestyle{plain}` `\thispagestyle{plain}`	richiede lo stile più semplice, in cui viene mostrato soltanto il numero di pagina;
`\pagestyle{empty}` `\thispagestyle{empty}`	richiede che venga eliminata qualunque indicazione, sopra e sotto il corpo del testo;
`\pagestyle{headings}` `\thispagestyle{headings}`	richiede che si utilizzino intestazioni e righe in fondo alla pagina in modo predefinito, secondo la classe del documento;
`\pagestyle{myheadings}` `\thispagestyle{myheadings}`	richiede che si utilizzino intestazioni e righe in fondo alla pagina secondo quanto richiesto espressamente con i comandi '**`\markboth`**' o '**`\markright`**';
`\markboth{`*intestazione_sinistra*`}`↩ ↪`{`*intestazione_destra*`}`	definisce delle righe di intestazione personalizzate, distinte per le pagine sinistre e per le pagine destre;
`\markright{`*intestazione_destra*`}`	definisce la riga di intestazione personalizzata per le pagine destre (si usa quando la classe del documento prevede di non distinguere tra pagine sinistre e destre);

Nell'elenco, sono stati messi assieme i comandi '**`\pagestyle`**' e '**`\thispagestyle`**', ma il primo interviene su tutte le pagine, a partire da quella in corso, fino al prossimo comando del genere, mentre il secondo riguarda esclusivamente la pagina in corso, come eccezione rispetto allo stile normale. LaTeX utilizza proprio questa tecnica per fare in modo che la pagina iniziale di un capitolo o di una parte sia diversa nello stile rispetto alle altre pagine normali.

Generalmente, le righe di intestazione sono definite automaticamente in modo appropriato, secondo la classe prescelta del documento. Per modificare manualmente queste righe di intestazione, si deve definire lo stile personalizzato, con il comando '**`\pagestyle{myheadings}`**', specificando il contenuto delle intestazioni con '**`\markboth`**' o con '**`\markright`**'. La definizione personalizzata delle intestazioni si limita a una riga di testo molto semplice, che viene allineata in modo appropriato (a sinistra nella pagine destre e a destra nelle pagine sinistre), per lasciare il posto al numero della pagina che viene inserito da LaTeX, senza poter intervenire su questo particolare.

A proposito dell'uso di '**`\markboth`**' e di '**`\markright`**', è interessante osservare che, se ci si trova in una pagina sinistra, viene preso in considerazione l'ultimo comando '**`\markboth`**' che rientri nella pagina, mentre se ci si trova in una pagina destra, viene preso in considerazione il primo comando '**`\markboth`**' o '**`\markright`**' che si trovi nella pagina stessa.

50.6.4 Numerazione delle pagine

LaTeX gestisce la numerazione delle pagine in modo automatico, attraverso il contatore '**`page`**'. A seconda della classe del documento, il contatore viene inizializzato in certi punti chiave, mostrando eventualmente una sequenza di numerazione differente. Se necessario, è possibile intervenire sul contatore e sull'aspetto della sequenza di numerazione.

Comando	Descrizione
`\pagenumbering{arabic}`	richiede una sequenza numerica normale (numeri arabi);
`\pagenumbering{roman}`	richiede una sequenza numerica con numeri romani fatti di lettere minuscole;
`\pagenumbering{Roman}`	richiede una sequenza numerica con numeri romani fatti di lettere maiuscole;
`\pagenumbering{alph}`	richiede una sequenza alfabetica minuscola;
`\pagenumbering{Alph}`	richiede una sequenza alfabetica maiuscola;
`\setcounter{page}{`***n***`}`	assegna alla pagina corrente il numero indicato; anche se si assegna un numero negativo, la progressione rimane in senso positivo.

Nella sezione 50.4.2 è stato descritto l'uso dei contatori; per modi-

ficare il contatore della pagina, è sufficiente agire sul nome '**page**', per esempio reinizializzandolo in questo modo:

```
\setcounter{page}{1}
```

Naturalmente, per intervenire in questo modo ci deve essere un buon motivo, perché la classe del documento provvede già a definire il modo migliore per reinizializzare il contatore al momento più opportuno.

Nello stesso modo, potrebbe essere interessante cambiare il modo in cui viene mostrato il numero della pagina, attraverso il comando '**\pagenumbering**', come descritto nella tabella.

50.6.5 Note a piè di pagina

Le note a piè di pagina vengono definite normalmente attraverso il comando '**\footnote**', che provvede da solo a incrementare il contatore delle note, corrispondente a '**footnote**', e a inserire la nota nella collocazione appropriata. Tuttavia, volendo fare riferimento alla stessa nota in più punti distinti, occorre separare la fase di dichiarazione da quella in cui si definisce il testo associato.

Comando	Descrizione
`\footnote{`*testo*`}`	incrementa il contatore delle note, inserisce il numero ad apice nel testo e mette la nota a piè di pagina;
`\footnotemark[\value{footnote}]`	lascia inalterato il contatore delle note e inserisce il numero attuale del contatore ad apice nel testo;
`\footnotemark[`*n*`]`	lascia inalterato il contatore delle note e inserisce il numero ***n*** ad apice nel testo;
`\footnotemark`	incrementa il contatore delle note, inserisce il numero ad apice nel testo, ma per la nota a piè di pagina attende il comando '**\footnotetext**';
`\footnotetext{`*testo*`}`	mette la nota a piè di pagina, associandole il numero attuale del contatore '**footnote**'.
`\footnotetext[`*n*`]{`*testo*`}`	mette la nota a piè di pagina, associandole il numero ***n***.

50.6.6 Sillabazione

Come per TeX, si può usare il comando '**\hyphenation**' per dichiarare la sillabazione di parole che in qualche modo non vengono separate correttamente; per esempio così:

```
\hyphenation{car-tel-lo-ne ma-nu-a-le}
```

Eventualmente, come già annotato in un'altra sezione, se c'è la necessità di mantenere unita una parola, o più parole assieme, si può usare una scatola:

```
\mbox{file system}
```

In questo caso, oltre che impedire la separazione delle parole, si vuole anche evitare che la riga venga interrotta in corrispondenza dello spazio che le separa.

50.7 Caratteri

In generale, LaTeX eredita la gestione dei caratteri da stampa del linguaggio TeX puro e semplice. In questo senso, valgono praticamente le stesse regole per ottenere le lettere accentate e gli altri simboli speciali, con la differenza che è possibile istruire LaTeX in modo che accetti la codifica UTF-8, con le istruzioni seguenti, da inserire nel preambolo, come descritto nella sezione 50.1.


```
\usepackage[utf8x]{inputenc}
\usepackage[T1]{fontenc}
```

Anche con LaTeX funzionano i legati in modo automatico, inoltre si definiscono i contesti matematici nello stesso modo di TeX.

La differenza importante di LaTeX sta nella definizione di una veste grafica complessiva e nel modo di fare riferimento ai caratteri da stampa, che avviene attraverso comandi differenti, con comportamenti diversi rispetto a TeX.

Negli esempi, salvo indicazione diversa, si suppone di utilizzare nel preambolo i comandi necessari all'utilizzo della codifica UTF-8:

```
\usepackage[utf8x]{inputenc}
\usepackage[T1]{fontenc}
```

50.7.1 Caratteri che hanno un significato speciale

Alcuni simboli (caratteri) usati nel sorgente hanno necessariamente un significato speciale, nel senso che non si traducono nel segno corrispondente nella composizione. Questo problema esiste già con TeX, ma LaTeX offre delle alternative leggermente diverse nel modo di ottenere i simboli mancanti nella composizione. La tabella 50.92 riepiloga i simboli che hanno un significato speciale nel sorgente LaTeX, con l'indicazione del codice necessario per ottenere i segni corrispondenti nella composizione.

Tabella 50.92. Elenco dei caratteri che hanno significati particolari.[2]

Carattere speciale	Utilizzo normale	Trasformazione per l'uso letterale
`\`	Prefisso di un comando o di un simbolo di controllo.	`\textbackslash`
`{` `}`	Apre e chiude un gruppo.	`\{` `\}`
`%`	Inizia un commento fino alla fine della riga.	`\%`
`&`	Tabulazione orizzontale.	`\&`
`~`	Spazio non interrompibile.	`\textasciitilde`
`$`	Inizia e conclude un contesto matematico.	`\$`
`^`	Apice, in un contesto matematico.	`\textasciicircum`
`_`	Il trattino basso rappresenta un pedice, in un contesto matematico.	`\_{}`
`#`	Definisce la collocazione di un parametro.	`\#`
`<SP>`	Uno o più spazi vengono ridotti a uno solo.	`\<SP>`
`[` `]`	Alle volte, le parentesi quadre delimitano un'opzione e non vengono inserite nella composizione.	`{[}` `{]}`

Per quanto riguarda gli accenti ottenuti sovrapponendo il simbolo alla lettera successiva, il funzionamento è uguale a quello di TeX, come descritto nelle sezioni 49.10.2 e 49.17.1.

50.7.2 Corpi uniformi

Lo scopo principale di LaTeX è quello di impostare una veste grafica uniforme, dove, tra le altre cose si specificano inizialmente i corpi e gli stili dei caratteri da stampa da usare nei vari contesti. In particolare, con il comando iniziale '**\documentclass**' è possibile indicare la dimensione normale del testo, attraverso un'opzione (come '**11pt**'), a cui si adegua proporzionalmente tutto il resto (sezione 50.1).

Alcuni comandi definiscono il corpo del carattere in base a delle definizioni relative:

Comando	Descrizione
{\Huge *testo*} \begin{Huge}*testo*\end{Huge}	il corpo più grande della serie;
{\huge *testo*} \begin{huge}*testo*\end{huge}	
{\LARGE *testo*} \begin{LARGE}*testo*\end{LARGE}	
{\Large *testo*} \begin{Large}*testo*\end{Large}	
{\large *testo*} \begin{large}*testo*\end{large}	
{\normalsize *testo*} \begin{normalsize} *testo* \end{normalsize}	la dimensione normale del testo;
{\small *testo*} \begin{small}*testo*\end{small}	
{\footnotesize *testo*} \begin{footnotesize} *testo* \end{footnotesize}	
{\scriptsize *testo*} \begin{scriptsize} *testo* \end{scriptsize}	
{\tiny *testo*} \begin{tiny}*testo*\end{tiny}	il corpo più piccolo nell'ambito della veste grafica scelta.

In condizioni normali, questi comandi non vengono usati direttamente da chi scrive un documento in LaTeX; tuttavia, se c'è la necessità di ingrandire o di rimpicciolire il testo, con questi c'è la certezza di rimanere nelle dimensioni standard usate dallo stile prescelto:

```
Questo è un testo normale, che può diventare più
{\large grande} o più {\small piccolo}, a seconda delle
esigenze.
```

Questo è un testo normale, che può diventare più grande o più piccolo, a seconda delle esigenze.

50.7.3 Caratteri da stampa

A differenza di TeX, i comandi di LaTeX che definiscono lo stile del carattere sono cumulabili. Per esempio, la definizione di un testo in corsivo può essere cumulata con quella del neretto, per ottenere un neretto corsivo.

Le caratteristiche principali di un carattere sono lo stile, la serie e il corpo. Lo stile e la serie possono essere definite utilizzando istruzioni del tipo seguente:

1. `\`***stile_serie_a***`{`***testo***`}`
2. `{\`***stile_serie_b***` `***testo***`}`
3. `\begin{`***stile_serie_b***`}`…`\end{`***stile_serie_b***`}`

I nomi delle istruzioni cambiano a seconda che si utilizzi un modo oppure l'altro. Per esempio, per scrivere un testo in corsivo, si possono utilizzare questi tre modi.

```
\textit{Testo in corsivo}

{\itshape Testo in corsivo}

\begin{itshape}Testo in corsivo\end{itshape}
```

Tutte le forme consentono di cumulare gli effetti; per esempio, si può scrivere '**`\textit{\textbf{`*testo*`}}`**', '**`{\itshape\bfseries`** ***testo*****`}`**', oppure '**`\begin{itshape}\begin{bfseries}`*testo*`\end{bfse`** per ottenere un neretto corsivo.

La tabella seguente elenca i comandi che possono essere usati in un contesto normale (non matematico):

Comando	Descrizione
\textrm{*testo*} {\rmfamily *testo*} \begin{rmfamily}*testo*\end{rmfamily}	tondo;
\textsf{*testo*} {\sffamily *testo*} \begin{sffamily}*testo*\end{sffamily}	lineare;
\texttt{*testo*} {\ttfamily *testo*} \begin{ttfamily}*testo*\end{ttfamily}	dattilografico;
\textbf{*testo*} {\bfseries *testo*} \begin{bfseries}*testo*\end{bfseries}	neretto;
\textup{*testo*} {\upshape *testo*} \begin{upshape}*testo*\end{upshape}	normale, in contrapposizione al neretto;
\textit{*testo*} {\itshape *testo*} \begin{itshape}*testo*\end{itshape}	corsivo;
\textsl{*testo*} {\slshape *testo*} \begin{slshape}*testo*\end{slshape}	inclinato;
\textsc{*testo*} {\scshape *testo*} \begin{scshape}*testo*\end{scshape}	maiuscoletto;
\textnormal{*testo*} {\normalfont *testo*} \begin{normalfont} *testo* \end{normalfont}	testo normale.

La tabella successiva riguarda invece il contesto matematico:

Comando	Descrizione
\mathnormal{*testo*}	testo matematico normale;
\mathrm{*testo*}	tondo;

Comando	Descrizione
`\mathbf{`*testo*`}`	neretto;
`\mathsf{`*testo*`}`	lineare;
`\mathtt{`*testo*`}`	dattilografico;
`\mathit{`*testo*`}`	corsivo;
`\mathcal{`*testo*`}`	calligrafico;
`\mathsc{`*testo*`}`	maiuscoletto.

A titolo di esempio, la tabella 50.99 consente di confrontare le varianti seriali dello stile normale di scrittura.

Tabella 50.99. Combinazioni a coppie di serie per il testo normale.

	tondo	corsivo	neretto	inclinato	maiuscoletto
tondo	abc123	*abc123*	**abc123**	*abc123*	ABC123
corsivo	*abc123*	*abc123*	***abc123***	*abc123*	ABC123
neretto	**abc123**	***abc123***	**abc123**	***abc123***	**ABC123**
inclinato	*abc123*	*abc123*	***abc123***	*abc123*	ABC123
maiuscoletto	ABC123	*abc123*	**ABC123**	ABC123	ABC123

Oltre allo stile e alla serie del carattere può essere definito il corpo, con i comandi già mostrati nella sezione precedente, oppure in modo diretto, nel caso si debba andare al di fuori della veste grafica preimpostata. In tal caso, si usano dei comandi nella forma:

```
\fontsize{corpo}{distanza_tra_le_righe}\selectfont
```

Per esempio, per ottenere un corpo di 5 mm, con un'altezza della riga di 6 mm (il 120 % del corpo), si può usare il comando '**`\fontsize{5mm}{6mm}\selectfont`**':

```
Questo è un testo normale; {\fontsize{5mm}{6mm}\selectfont
questo carattere ha un corpo particolare}; qui il testo
torna a essere normale.
```

Questo è un testo normale; questo carattere ha un corpo particolare; qui il testo torna a essere normale.

50.7.3.1 Enfatizzazione

Quando non ha importanza stabilire esattamente l'aspetto di un carattere, ma si vuole semplicemente enfatizzarlo rispetto al contesto in cui si trova, si può usare il comando '**`\emph`**'. L'esempio seguente mostra cosa accade quando in una situazione in cui il carattere è corsivo si usa questo comando:

```
\begin{itshape}Bla bla bla bla bla bla \emph{testo
enfatizzato} bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla...\end{itshape}
```

Bla bla bla bla bla bla testo enfatizzato *bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla...*

50.7.4 Definizione dettagliata del carattere da stampa

Generalmente, i comandi già descritti a proposito della definizione del carattere da stampa sono più che sufficienti per l'utente normale di LaTeX; eventualmente, sono disponibili anche altri comandi più dettagliati, che si prestano particolarmente per la realizzazione di macro particolari. Questi comandi hanno in comune la necessità di usare alla fine il comando '**`\selectfont`**' per attuare le modifiche al carattere da stampa. La tabella seguente elenca i comandi che riguardano la definizione dettagliata del carattere da stampa (appare anche '**`\fontsize`**' che è già descritto nella sezione precedente):

Comando	Descrizione
`\fontencoding{OT1|T1|`*altro_insieme*`}`	definisce l'insieme di caratteri ('**`T1`**' rappresenta quello utile per le lingue europee occidentali);

Comando	Descrizione
`\fontfamily{cmr|cmss|cmtt|`*altro_stile*`}`	definisce lo stile del carattere ('**`cmr`**' sta per *Computer modern roman*, '**`cmss`**' è il *Computer modern sans serifs*, '**`cmtt`**' è il *Computer modern typewriter text*);
`\fontseries{m|b|c|bc|bx|`*altra_serie*`}`	definisce la serie del carattere ('**`m`**' sta per *Medium*, '**`b`**' sta per *Bold*, '**`c`**' sta per *Condensed*, '**`bc`**' sta per *Bold condensed* e '**`bx`**' sta per *Bold extended*);
`\fontshape{n|it|sl|sc|ui|ol}`	definisce una variante della serie del carattere ('**`n`**' sta per *Normal*, '**`it`**' sta per *Italic*, '**`sl`**' sta per *Slanted*, '**`sc`**' sta per *Small caps*, '**`ui`**' sta per *Upright italic* e '**`ol`**' sta per *Outline*, ma naturalmente non tutti i tipi di carattere hanno a disposizione tutte le varianti possibili);
`\fontsize{`*corpo*`}{`*altezza_riga*`}`	definisce il corpo del carattere e l'altezza della riga;
`\selectfont`	abilita le modifiche fatte in precedenza al carattere (tipo, stile e dimensione);
`\usefont{`*insieme*`}{`*stile*`}{`*forma*`}`	definisce simultaneamente tutte le caratteristiche del carattere, escluso il corpo e la distanza tra le righe, senza nemmeno bisogno di '**`\selectfont`**' per attivare la scelta fatta.

A proposito del comando '**`\usefont`**', si osservi che il secondo argomento si riferisce alle sigle utilizzabili con il comando '**`\fontseries`**', mentre il terzo argomento richiede una sigla riferita al comando '**`\fontshape`**'. La suddivisione delle varianti seriali in questi due gruppi deriva dall'organizzazione di LaTeX riguardo alla cumulabilità tra stili. In pratica, le varianti seriali di '**`\fontseries`**' non sono cumulabili tra di loro; per esempio, per ottenere un carattere stretto e neretto è già prevista una voce unica: '**`bc`**'. Lo stesso ragionamento vale per le varianti seriali di '**`\fontshape`**', mentre una variante di '**`\fontseries`**' è cumulabile con una variante di '**`\fontshape`**'.

Viene mostrato un esempio completo, in cui si può notare anche l'uso delle lettere accentate, a seguito della selezione dell'insieme '**`T1`**':

```
\documentclass{article}
\usepackage[italian]{babel}
%\usepackage[utf8x]{inputenc}
%\usepackage[T1]{fontenc}
\frenchspacing
\pagestyle{empty}
\setlength{\textwidth}{12cm}
\begin{document}

Questo \'e un testo normale;
{\fontencoding{T1}\fontfamily{cmss}\fontseries{n}
\fontshape{it}\fontsize{5mm}{7mm}\selectfont questo \'e
invece un carattere senza grazie, chiaro, corsivo, con un
corpo di 5 mm, in una riga alta 7 mm.} Questo \'e di nuovo
un carattere normale. \usefont{T1}{cmss}{n}{it}Questo \'e
ancora un carattere senza grazie, chiaro, corsivo, con un
corpo normale. \fontsize{5mm}{7mm}\selectfont Qui il
carattere si ingrandisce di nuovo. \normalfont Qui si
ritorna a un carattere di stile e serie normale, ma la
dimensione rimane inalterata.

\end{document}
```

Questo è un testo normale; questo è invece un carattere senza grazie, chiaro, corsivo, con un corpo di 5 mm, in una riga alta 7 mm. Questo è di nuovo un carattere normale. Questo è ancora un carattere senza grazie, chiaro, corsivo, con un corpo normale. Qui il carattere si ingrandisce di nuovo. Qui si ritorna a un carattere di stile e serie normale, ma la dimensione rimane inalterata.

50.7.5 Linee e cornicette attorno al testo

Per LaTeX vale lo stesso problema di TeX a proposito della sottolineatura, nel senso che non fa parte dello stile e della serie del carattere. Tuttavia, LaTeX facilita anche altre forme di evidenziamento analoghe:

Comando	Descrizione
`\underline{`*testo*`}`	mette una linea sotto al testo;
`$\overline{\hbox{`*testo*`}}$`	mette una linea sopra al testo;
`\frame{`*testo*`}`	racchiude il testo in un rettangolo;
`\fbox{`*testo*`}`	racchiude il testo in un rettangolo, lasciando un po' di spazio.

In tutti i casi, il testo evidenziato si trova inserito in una scatola orizzontale, che come tale non può essere spezzata, creando eventualmente dei problemi di impaginazione.

Si osservi che il comando '**\overline**' è fatto per l'ambito matematico. Nella tabella, la sintassi mostra il modo corretto per poterlo utilizzare in un contesto normale, con l'aggiunta del comando '**\hbox**', allo scopo di comporre il testo sopralineato in un contesto normale.

Segue un esempio riepilogativo di queste forme di evidenziamento del testo:

```
Qui inizia un testo normale, in cui questa
\underline{parola} appare sottolineata; ma è possibile
anche sopralineare $\overline{\hbox{questa
piccola frase}}$, così come si possono creare delle
\frame{cornicette schiacciate}, oppure \fbox{un po' più
distanziate} dal testo.
```

Qui inizia un testo normale, in cui questa parola appare sottolineata; ma è possibile anche sopralineare questa piccola frase, così come si possono creare delle cornicette schiacciate, oppure un po' più distanziate dal testo.

50.7.6 Spostamento verticale del testo

A volte può essere necessario lo spostamento verticale di alcuni caratteri rispetto alla base della riga. Per fare questo si usa il comando '**\raisebox**', con cui si delimita il testo da spostare, specificando di quanto questo debba essere alzato (per abbassarlo si possono usare valori negativi):

```
\raisebox{spostamento_verticale}{testo}
```

Si osservi l'esempio seguente in cui si gioca con la parola «ciao»:

```
c\raisebox{1mm}{i}\raisebox{2mm}{a}\raisebox{1mm}{o}
c\raisebox{-1mm}{i}\raisebox{-2mm}{a}\raisebox{-1mm}{o}
```

$$c^{i^{a}o}\ c_{i_{a}o}$$

50.8 LaTeX: blocchi di testo

LaTeX impagina il testo in paragrafi, in modo analogo a quanto fa già il linguaggio TeX puro e semplice. Anche per LaTeX funziona il comando '**\par**', allo scopo di concludere un paragrafo in modo esplicito, oppure è sufficiente lasciare almeno una riga bianca tra un blocco e l'altro.

50.8.1 Caratteristiche normali di un paragrafo

Il paragrafo di LaTeX è costruito in maniera molto simile a quella di TeX, con la differenza che il controllo sulla sua estensione orizzontale dipende dall'impostazione della pagina e dalla suddivisione in colonne. Anche con LaTeX rimane il rientro della prima riga dei paragrafi, con l'eccezione normale del primo paragrafo dopo un titolo. La tabella 50.112 mostra alcuni dei comandi che controllano l'aspetto del paragrafo.

Tabella 50.112. Alcuni dei comandi che intervengono nella definizione del paragrafo.

Lunghezza	Valore predefinito	Descrizione
`\parindent`	0.0pt	Larghezza del rientro della prima riga (in questo caso il contesto è quello di una tabella; di solito il valore è maggiore di zero).
`\parskip`	0.0pt	Distanza aggiuntiva tra i paragrafi.
`\linewidth`	79.66649pt	Larghezza della riga (in condizioni normali coincide con la larghezza della colonna a disposizione).
`\baselineskip`	12.0pt	Distanza verticale tra la base di una riga e la base di quella successiva (questo valore viene modificato automaticamente se si cambia la dimensione del carattere da stampa).
`\leftskip`	0.0pt	Rientro sinistro del paragrafo.
`\rightskip`	0.0pt	Rientro destro del paragrafo.

Senza intervenire nella lunghezza definita dal comando '**\parindent**', è possibile annullare il rientro della prima riga con il comando '**\noindent**', così come è possibile usare '**\indent**' per ottenere l'effetto opposto in presenza di un paragrafo che non ha il rientro per qualche ragione. Tuttavia, anche annullando il rientro con il comando '**\noindent**', rimane il problema della distanza tra i paragrafi, che in condizioni normali risulta troppo piccolo per consentirne l'individuazione senza rientri. Pertanto, quando si intende scrivere un documento che non fa uso di rientri in modo sistematico, conviene modificare le lunghezze che contano, direttamente nel preambolo, per esempio così:

```
\setlength{\parindent}{0pt}
\setlength{\parskip}{1ex plus 0.5ex minus 0.2ex}
```

Si osservi però, che in questo modo non può funzionare nemmeno il comando '**\indent**', perché si limiterebbe a porre un rientro di zero punti.

Segue il riepilogo degli ultimi comandi descritti:

Comando	Descrizione
`\indent`	richiede espressamente il rientro della prima riga secondo il valore di '**\parindent**';
`\noindent`	annulla il rientro della prima riga per il paragrafo in cui si trova.

50.8.2 Allineamento del testo

In condizioni normali, il testo dei paragrafi risulta allineato simultaneamente a sinistra e a destra. Per modificare questo comportamento si interviene normalmente attraverso degli ambienti, ma sono disponibili anche dei comandi alternativi che provengono da TeX:

Comando	Descrizione
`\begin{center}`*testo*`\end{center}`	centra il testo;
`\centerline{`*riga_di_testo*`}`	centra la riga;
`\begin{flushleft}`*testo*`\end{flushleft}` oppure `{\raggedright` *testo*`\par}`	allinea il testo a sinistra;
`\begin{flushright}` *testo* `\end{flushright}` oppure `{\raggedleft` *testo*`\par}`	allinea il testo a destra.

È interessante osservare che, mentre i comandi '**\raggedright**' e '**\raggedleft**' richiedono l'inserimento in un raggruppamento, dove l'ultimo paragrafo deve terminare in modo esplicito, prima della fine del raggruppamento, gli ambienti dichiarano implicitamente l'inizio e la fine di un paragrafo.

Il comando '**\centerline**' consente di centrare una riga soltanto, senza inserire la spaziatura che riguarda un paragrafo normale.

```
Questo è un paragrafo normale, che serve come riferimento.
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.

\begin{center}
Questo è un paragrafo centrato, attraverso l'ambiente
{\ttfamily center}. Bla bla bla bla bla bla bla bla bla
bla bla.
\end{center}

Questo è un paragrafo normale, a cui segue una riga
centrata:

\centerline{riga centrata.}

\begin{flushleft}
Questo è un paragrafo allineato a sinistra, attraverso
l'ambiente {\ttfamily flushleft}. Bla bla bla bla bla bla
bla bla bla bla bla.
\end{flushleft}

\begin{flushright}
Questo è un paragrafo allineato a destra, attraverso
l'ambiente {\ttfamily flushright}. Bla bla bla bla bla
bla bla bla bla bla.
\end{flushright}
```

Questo è un paragrafo normale, che serve come riferimento. 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.

Questo è un paragrafo centrato, attraverso l'ambiente `center`. Bla bla bla bla bla bla bla bla bla bla bla.

Questo è un paragrafo normale, a cui segue una riga centrata:
riga centrata.

Questo è un paragrafo allineato a sinistra, attraverso l'ambiente `flushleft`. Bla bla bla bla bla bla bla bla bla bla bla.

Questo è un paragrafo allineato a destra, attraverso l'ambiente `flushright`. Bla bla bla bla bla bla bla bla bla bla bla.

50.8.3 Elenchi

«

Alcuni ambienti sono predisposti per la realizzazione di elenchi di vario tipo. Quelli più comuni sono '**description**', '**enumerate**' e '**itemize**', ma esiste anche il modo di definire un tipo di elenco libero, attraverso l'ambiente generico '**list**'. L'elenco puntato classico si ottiene con l'ambiente '**itemize**':

```
\begin{itemize}
\item testo
\item testo
…
\item testo
\end{itemize}
```

All'interno dell'ambiente '**itemize**', il comando '**\item**' serve a indicare l'inizio di una voce dell'elenco; la voce può contenere del testo piuttosto articolato, anche in più paragrafi, per esempio con dei sottoelenchi o ambienti di altro genere. L'esempio seguente mostra un elenco contenente un sottoelenco:

```
\begin{itemize}
\item Primo.
\item Secondo.
      \begin{itemize}
      \item Questa voce è una suddivisione ulteriore del
            secondo punto.

            Questo è un paragrafo ulteriore.
      \item Questa è una voce ulteriore della
            sottoclassificazione appartenente al secondo
            punto.
      \end{itemize}
\item Terzo.
\end{itemize}
```


- Primo.
- Secondo.
  - Questa voce è una suddivisione ulteriore del secondo punto.
    Questo è un paragrafo ulteriore.
  - Questa è una voce ulteriore della sottoclassificazione appartenente al secondo punto.
- Terzo.

Teoricamente, è possibile modificare il simbolo usato per evidenziare l'inizio delle voci, ridefinendo i comandi '**\labelitemi**', '**\labelitemii**', '**\labelitemiii**' e '**\labelitemiv**', che si riferiscono rispettivamente all'elenco principale e di seguito a sottoelenchi fino a un massimo di quattro livelli di annidamento. Per esempio, si può fare in modo che si utilizzi il segno «+» per l'elenco principale intervenendo in questo modo:

```
\renewcommand{\labelitemi}{\normalfont +}
```

Ecco come si presenta l'elenco già mostrato se si aggiunge questa istruzione all'inizio:

+ Primo.

+ Secondo.
  - Questa voce è una suddivisione ulteriore del secondo punto.
    Questo è un paragrafo ulteriore.
  - Questa è una voce ulteriore della sottoclassificazione appartenente al secondo punto.

+ Terzo.

In modo analogo si ottengono degli elenchi numerati, con l'ambiente '**enumerate**':

```
\begin{enumerate}
\item testo
\item testo
…
\item testo
\end{enumerate}
```

L'esempio già visto viene riproposto usando l'ambiente '**enumerate**'; si può osservare che la numerazione delle voci cambia quando si inseriscono dei sottoelenchi:

```
\begin{enumerate}
\item Primo.
\item Secondo.
      \begin{enumerate}
      \item Questa voce è una suddivisione ulteriore del
            secondo punto.

            Questo è un paragrafo ulteriore.
      \item Questa è una voce ulteriore della
            sottoclassificazione appartenente al secondo
            punto.
      \end{enumerate}
\item Terzo.
\end{enumerate}
```

1. Primo.
2. Secondo.
   (a) Questa voce è una suddivisione ulteriore del secondo punto.
   Questo è un paragrafo ulteriore.
   (b) Questa è una voce ulteriore della sottoclassificazione appartenente al secondo punto.
3. Terzo.

La sequenza dei valori attribuita alle voci viene determinata attra-

verso dei contatori, che sono rispettivamente: '**enumi**', '**enumii**', '**enumiii**' e '**enumiv**'. In pratica, il primo contatore controlla l'elenco più esterno, mentre l'ultimo riguarda quello più interno (di conseguenza ci possono essere un massimo di quattro annidamenti). Inoltre, il tipo di numerazione usata negli elenchi dipende rispettivamente dai comandi: '**\theenumi**', '**\theenumii**', '**\theenumiii**' e '**\theenumiv**'. L'esempio seguente è una variante di quello già visto, in cui, l'elenco più esterno parte dal numero tre, mentre l'elenco annidato utilizza i numeri romani:

```
\begin{enumerate}
\setcounter{enumi}{2}
\item Primo.
\item Secondo.
      \renewcommand{\theenumii}{\roman{enumii}}
      \begin{enumerate}
      \item Questa voce è una suddivisione ulteriore del
            secondo punto.

            Questo è un paragrafo ulteriore.
      \item Questa è una voce ulteriore della
            sottoclassificazione appartenente al secondo
            punto.
      \end{enumerate}
\item Terzo.
\end{enumerate}
```

3. Primo.

4. Secondo.

   (i) Questa voce è una suddivisione ulteriore del secondo punto.
   Questo è un paragrafo ulteriore.

   (ii) Questa è una voce ulteriore della sottoclassificazione appartenente al secondo punto.

5. Terzo.

Gli elenchi descrittivi sono leggermente diversi, perché devono consentire la descrizione delle voci. L'ambiente usato per questo è '**description**':

```
\begin{description}
\item [etichetta] testo
\item [etichetta] testo
…
\item [etichetta] testo
\end{description}
```

Ciò che è stato indicato come ***etichetta*** è ciò che rappresenta simbolicamente la voce che viene descritta dal testo successivo; come si vede dalla sintassi, potrebbe essere omessa, anche se questo fa perdere di significato a un elenco descrittivo. Viene riproposto l'esempio già visto in precedenza per gli altri tipi di elenco, con l'aggiunta delle etichette:

```
\begin{description}
\item [marrone] Primo.
\item [rosso] Secondo.
      \begin{description}
      \item [oro] Questa voce è una suddivisione ulteriore
            del secondo punto.

            Questo è un paragrafo ulteriore.
      \item [argento] Questa è una voce ulteriore della
            sottoclassificazione appartenente al secondo
            punto.
      \end{description}
\item [arancio] Terzo.
\end{description}
```

marrone Primo.

rosso Secondo.

oro Questa voce è una suddivisione ulteriore del secondo punto.
Questo è un paragrafo ulteriore.

argento Questa è una voce ulteriore della sottoclassificazione appartenente al secondo punto.

arancio Terzo.

Come si vede dal risultato della composizione, l'annidamento non fa cambiare l'aspetto dell'elenco.

Eventualmente, si può omettere l'indicazione della voce distintiva dei vari punti, se a questo si provvede da soli con gli evidenziamenti che si preferiscono; tuttavia, può risultare difficile inserire le parentesi quadre all'inizio di queste voci. Per inserire le parentesi quadre con un significato letterale, basta fare in modo che LaTeX comprenda che il testo è già iniziato, attraverso l'inserzione di gruppi vuoti, oppure inserendo le parentesi stesse tra parentesi graffe:

```
\begin{description}
\item [marrone] Primo.
\item {[rosso]} Secondo.
\item {}[arancio] Terzo.
\end{description}
```

marrone Primo.

[rosso] Secondo.

[arancio] Terzo.

Gli ambienti riferiti agli elenchi gestiscono in modo autonomo la distanza verticale di separazione tra le voci, impostando internamente il comando '**\itemsep**'. Volendo, si può interferire in questa lunghezza per modificare la distanza tra le voci di un elenco, ma solo all'interno dell'elenco stesso:

```
\begin{itemize}
\item Primo. In questo contesto, il valore predefinito di
      {\ttfamily \textbackslash{}itemsep} è \the\itemsep{}.
\setlength{\itemsep}{1cm}
\item Secondo.
      \begin{itemize}
      \item Questa voce è una suddivisione ulteriore del
            secondo punto.

            Questo è un paragrafo ulteriore.
      \item Questa è una voce ulteriore della
            sottoclassificazione appartenente al secondo
            punto.
      \end{itemize}
\item Terzo.
\end{itemize}
```

- Primo. In questo contesto, il valore predefinito di `\itemsep` è 4.0pt plus 2.0pt minus 1.0pt.

- Secondo.

  – Questa voce è una suddivisione ulteriore del secondo punto.
  Questo è un paragrafo ulteriore.

  – Questa è una voce ulteriore della sottoclassificazione appartenente al secondo punto.

- Terzo.

Infine, si può realizzare un elenco in modo libero, attraverso l'ambiente '**list**':

```
\begin{list}{etichetta}{comandi_di_spaziatura}
\item testo
\item testo
…
\item testo
\end{list}
```

In pratica, al posto di ***etichetta*** si inseriscono i comandi necessari a produrre l'etichetta che contrassegna le voci dell'elenco, mentre al posto di ***comandi_di_spaziatura*** si possono mettere i comandi che servono a definire le lunghezze relative alla costruzione dell'elenco. Per cominciare, conviene mostrare un esempio banale di elenco puntato, dove il punto delle voci è rappresentato da un asterisco:

```
\begin{list}{*}{}
\item Primo.
\item Secondo.
\item Terzo.
\end{list}
```

* Primo.

* Secondo.

* Terzo.

Per controllare la distanza tra i punti, si può modificare la distanza '**`\itemsep`**':

```
\begin{list}{*}{\setlength{\itemsep}{1cm}}
\item Primo.
\item Secondo.
\item Terzo.
\end{list}
```

* Primo.

* Secondo.

* Terzo.

Per costruire un elenco numerato, occorre intervenire con maggiore abilità nella definizione dell'etichetta:

```
\newcounter{elenco}

\setcounter{elenco}{0}
\begin{list}{\stepcounter{elenco}\arabic{elenco}}{\setlength{\itemsep}{1cm}}
\item Primo.
\item Secondo.
\item Terzo.
\end{list}
```

1 Primo.

2 Secondo.

3 Terzo.

50.8.4 Citazioni

«

Sono disponibili due ambienti per rappresentare facilmente del testo rientrato rispetto al flusso normale:

Comando	Descrizione
`\begin{quote}`***testo***`\end{quote}` `\begin{quotation}`***testo***`\end{quotation}`	rientra a destra e a sinistra il testo, in un modo conveniente per includere una citazione.

La differenza tra i due ambienti sta nel fatto che '**`quote`**' non fa rientrare la prima riga dei paragrafi, mentre '**`quotation`**' si comporta come al solito. In ogni caso, entrambi gli ambienti possono contenere più paragrafi o blocchi di altro tipo. Segue un esempio:

```
Bla bla bla bla bla...

\begin{quote}

    This program is free software; 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.

\end{quote}

Bla bla bla bla bla...

\begin{quotation}

    This program is distributed in the hope that it will be
    useful, but WITHOUT ANY WARRANTY; without even the
    implied warranty of MERCHANTABILITY or FITNESS FOR A
    PARTICULAR PURPOSE.  See the GNU General Public License
    for more details.

    You should have received a copy of the GNU General
    Public License along with this program; if not, write
    to the Free Software Foundation, Inc., 675 Mass Ave,
    Cambridge, MA 02139, USA.

\end{quotation}
```


Bla bla bla bla bla...

> This program is free software; 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.

Bla bla bla bla bla...

> This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.
> You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.

50.8.5 Versi

«

LaTeX facilita la scrittura di versi poetici attraverso l'ambiente '**`verse`**'. Si osservi l'esempio seguente:

```
\section{La Divina Commedia -- Inferno}

canto XV

\begin{verse}
Ora c'en porta l'un de' duri margini;\par
e 'l fummo del ruscel di sopra aduggia,\par
sì che dal foco salva l'acqua e li argini.\par
\end{verse}
```

1 La Divina Commedia – Inferno

canto XV

Ora c'en porta l'un de' duri margini;
e 'l fummo del ruscel di sopra aduggia,
sì che dal foco salva l'acqua e li argini.

Come si può osservare, le strofe devono occupare ognuna un paragrafo separato. L'ambiente '**`verse`**' si occupa in pratica di rientrare in modo particolare i paragrafi, per consentire di riconoscere anche le strofe lunghe. Si osservi cosa accade con una strofa più lunga dello spazio che può avere a disposizione la riga:

prima strofa;

seconda strofa, molto lunga, 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 bla.

50.8.6 Testo letterale

«

Per riprodurre del testo in modo letterale, è disponibile l'ambiente '**`verbatim`**' e il comando '**`\verb`**':

Comando	Descrizione

`\begin{verbatim}`
testo_preformattato
`\end{verbatim}`
	compone il testo in modo letterale, rispettando anche le interruzioni di riga, usando un carattere dattilografico;

`\begin{verbatim*}`
testo_preformattato
`\end{verbatim*}`
	come nel caso precedente, evidenziando gli spazi con un segno speciale.

`\verb`***x******testo_letterale******x***	compone il testo, che non può superare la riga di lunghezza, in modo letterale, usando un carattere dattilografico;

Comando	Descrizione
`\verb*`*x*`testo_letterale`*x*	come nel caso precedente, evidenziando gli spazi con un segno speciale.

Come si può comprendere dalla spiegazione data nella tabella riepilogativa, l'ambiente '**verbatim**' consente di rappresentare facilmente dei blocchi di testo letterali, da riprodurre così come sono nel sorgente, anche nelle interruzioni di riga; al contrario, il comando '**\verb**' va usato nel testo normale.

Nel caso dell'ambiente '**verbatim**', il contenuto del testo non ha limitazioni, salvo il fatto che non può includere il comando di chiusura dell'ambiente ('**\end{verbatim}**'); mentre il comando '**\verb**' delimita il testo letterale con un carattere a scelta, inserito all'inizio e alla fine del testo. Gli esempi seguenti mostrano l'uso dei due modi di indicare testo letterale, in modo normale o con l'asterisco:

```
\$ \verb@cat -n /bin/bash@ [Invio]

\begin{verbatim}
     1  ^?ELF^A^A^A^@^@^@^@^@^@^@^@^@^B^@^C^@^A^@^@^@M-PM-^Q
     2  ^F^@^@M-w^B^@^@^_^F^@^@M-^^^D^@^@^@^@^@^@^^^F^@^@^K^
     3  ^D^@^@b^F^@^@^@^@^@^@_^C^@^@M-^[^C^@^@M-^[^@^@^@M-^J
    ...
\end{verbatim}
```

$ cat -n /bin/bash [Invio]

```
     1  ^?ELF^A^A^A^@^@^@^@^@^@^@^@^@^B^@^C^@^A^@^@^@M-PM-^Q
     2  ^F^@^@M-w^B^@^@^_^F^@^@M-^^^D^@^@^@^@^@^@^^^F^@^@^K^
     3  ^D^@^@b^F^@^@^@^@^@^@_^C^@^@M-^[^C^@^@M-^[^@^@^@M-^J
...
```

```
\$ \verb*@ls -l /@ [Invio]

\begin{verbatim*}
drwxr-xr-x    2 root     root         4096 mag  3 22:46 bin
drwxr-xr-x    4 root     root         4096 apr 20 16:15 boot
drwxr-xr-x    7 root     root        20480 mag  5 08:49 dev
drwxr-xr-x  123 root     root         8192 mag  5 08:49 etc
drwxr-xr-x   21 root     root         4096 feb 10 20:59 home
drwxr-xr-x    7 root     root         8192 mag  3 22:48 lib
...
\end{verbatim*}
```

$ ls␣-l␣/ [Invio]

```
drwxr-xr-x␣␣␣␣2␣root␣␣␣␣␣root␣␣␣␣␣␣␣␣4096␣mag␣␣3␣22:46␣bin
drwxr-xr-x␣␣␣␣4␣root␣␣␣␣␣root␣␣␣␣␣␣␣␣4096␣apr␣20␣16:15␣boot
drwxr-xr-x␣␣␣␣7␣root␣␣␣␣␣root␣␣␣␣␣␣␣␣20480␣mag␣␣5␣08:49␣dev
drwxr-xr-x␣␣123␣root␣␣␣␣␣root␣␣␣␣␣␣␣␣8192␣mag␣␣5␣08:49␣etc
drwxr-xr-x␣␣␣21␣root␣␣␣␣␣root␣␣␣␣␣␣␣␣4096␣feb␣10␣20:59␣home
drwxr-xr-x␣␣␣␣7␣root␣␣␣␣␣root␣␣␣␣␣␣␣␣8192␣mag␣␣3␣22:48␣lib
...
```

Si veda anche la sezione 50.14.2 per l'importazione di file esterni da rappresentare in modo letterale.

50.8.7 Ambienti fluttuanti

LaTeX dispone di due ambienti fluttuanti nel testo, preparati idealmente per incorporare tabelle o figure:

```
\begin{figure}[permesso_di_collocazione]
    figura
\end{figure}
```

```
\begin{table}[permesso_di_collocazione]
    tabella
\end{table}
```

Per il momento, per come sono mostrati i modelli sintattici dell'utilizzo di questi ambienti, non c'è differenza tra un ambiente o l'altro. La collocazione dell'oggetto contenuto nell'ambiente fluttuante, viene concessa attraverso l'uso di alcune lettere alfabetiche, con il significato seguente:

Lettera	Significato mnemonico	Descrizione
`h`	*here*	si concede la collocazione nel punto in cui si trova la dichiarazione della tabella o della figura;
`t`	*top*	si concede la collocazione all'inizio della pagina attuale;
`b`	*bottom*	si concede la collocazione alla fine della pagina attuale;
`p`	*page*	si concede la collocazione in una pagina a parte, riservata a questo.

Queste lettere possono essere sommate assieme e conta l'ordine in cui sono dati i permessi di collocazione. Per esempio,

```
\begin{table}[tbp]
    …
\end{table}
```

dichiara una tabella fluttuante che può essere collocata nell'ordine: all'inizio della pagina, alla fine, oppure in una pagina a parte. In pratica, viene esclusa la collocazione nel punto in cui viene dichiarata, a meno che questo coincida con una delle alternative concesse.

Se i permessi di posizionamento dell'oggetto non vengono specificati, LaTeX intende l'uso della sigla '**tbp**', cioè in alto, in basso o in una nuova pagina, escludendo la posizione naturale.

Dal momento che un oggetto inserito in un ambiente fluttuante non può essere suddiviso, risulta difficile imporre la collocazione nello stesso punto in cui si trova, quando ciò costringe a lasciare molto spazio vuoto alla fine di una pagina. Pertanto, l'uso del solo permesso '**h**', senza alternative, diventa privo del suo significato nella maggior parte dei casi. Se la collocazione di questi oggetti deve avvenire necessariamente nella loro posizione naturale, lo si può imporre incorporando il pacchetto '**float**' nel preambolo e indicando il permesso '**H**' (la lettera H maiuscola):

```
\usepackage{float}
\begin{document}
…
\begin{table}[H]
    …
\end{table}
```

La differenza tra gli ambienti si avverte solo quando si inseriscono le didascalie, attraverso il comando '**\caption**':

```
\begin{figure}[permesso_di_collocazione]
    \caption{didascalia}
    figura
\end{figure}
```

```
\begin{table}[permesso_di_collocazione]
    \caption{didascalia}
    tabella
\end{table}
```

Il comando '**\caption**' può essere inserito in qualunque punto, all'interno dell'ambiente, purché ciò possa avere senso in fase di composizione (generalmente si sceglie l'inizio o la fine, per avere una didascalia superiore o inferiore). L'inserimento del comando fa incrementare il contatore delle figure o delle tabelle, corrispondente rispettivamente a '**figure**' o '**table**'; inoltre la didascalia risulta arricchita della definizione «figura» o «tabella» con l'aggiunta del numero relativo. Infine, il numero della figura o della tabella viene accumulato per generare un indice specifico, attraverso i comandi '**\listoffigures**' e '**\listoftables**'.

L'esempio seguente mostra la costruzione di una tabella molto povera, che serve a comprendere quanto si indifferente per LaTeX cosa sia il contenuto effettivo di questi ambienti:

```
\begin{table}[htbp]
    \begin{verbatim}
            .-------------------------------------------.
            | n. | cognome  | nome          | professione |
            |----|----------|---------------|-------------|
            |  1 | Tizi     | Tizio         | tramviere   |
            |----|----------|---------------|-------------|
            |  2 | Cai      | Caio          | pasticcere  |
            |----|----------|---------------|-------------|
            |  3 | Semproni | Sempronio     | falegname   |
            `-------------------------------------------'
    \end{verbatim}
    \caption{Una tabella molto semplice}
\end{table}
```

```
.-------------------------------------------.
| n. | cognome  | nome          | professione |
|----|----------|---------------|-------------|
|  1 | Tizi     | Tizio         | tramviere   |
|----|----------|---------------|-------------|
|  2 | Cai      | Caio          | pasticcere  |
|----|----------|---------------|-------------|
|  3 | Semproni | Sempronio     | falegname   |
`-------------------------------------------'
```

Tabella 1: Una tabella molto semplice

Nella sezione 50.6.2 sono descritti i comandi con cui si può ottenere un salto pagina. In particolare, i comandi '**`\clearpage`**' e '**`\cleardoublepage`**' si occupano anche di inserire gli oggetti fluttuanti che non fossero già stati collocati secondo i permessi attribuiti. In pratica, se degli oggetti fluttuanti non hanno ancora potuto essere collocati, questi comandi per il salto pagina forzano il loro l'inserimento nella composizione, probabilmente in una pagina a parte. Di solito, i comandi che in qualche modo implicano un salto pagina, fanno uso di questi comandi '**`\clearpage`**' o '**`\cleardoublepage`**', proprio per chiudere anche con gli oggetti fluttuanti ancora sospesi.

50.9 Spazi e scatole

LaTeX evita di dare la stessa libertà che dà TeX nella gestione degli spazi e delle scatole. In questo senso, sono disponibili una serie di comandi, spesso simili tra loro, che possono creare un po' di confusione all'utilizzatore.

50.9.1 Spazi orizzontali

A proposito di spazi orizzontali, esistono ovviamente due possibilità: spazi rigidi e spazi elastici. Nel secondo caso, sono disponibili anche comandi che inseriscono puntini o linee in modo elastico, come quando si compila un indice e si vuole guidare la vista verso il numero della pagina.

Comando	Descrizione
`\hspace{`***lunghezza***`}` `\hspace*{`***lunghezza***`}`	Inserisce uno spazio orizzontale rigido. Il valore dell'argomento può essere negativo; in tal caso si ottiene un arretramento. Se lo spazio inserito viene a trovarsi all'inizio di una riga, viene ignorato, a meno che sia usato l'asterisco.

L'esempio seguente vuole mettere a confronto uno spazio normale, che viene soppresso perché si trova all'inizio di una riga, con uno spazio con asterisco, che risulta inserito in ogni caso:

```
1 Bla bla bla \hspace{3cm} bla bla bla...

2 Bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla \hspace{5cm} bla bla
bla...

3 Bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla \hspace*{5cm} bla bla
bla...
```


1 Bla bla bla   bla bla bla...
2 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...
3 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...

Comando	Descrizione
`\hfill`	Inserisce uno spazio orizzontale elastico che, in condizioni normali, va a riempire tutto lo spazio disponibile.
`\dotfill`	Inserisce uno spazio orizzontale elastico contenente dei puntini.
`\hrulefill`	Inserisce uno spazio orizzontale elastico contenente una linea orizzontale.

Generalmente, lo spazio orizzontale elastico si usa per allineare qualcosa al centro o a destra, come nell'esempio seguente:

```
\noindent sinistra\hfill centro\hfill destra\par
\noindent sinistra\dotfill centro\hrulefill destra\par
```

Come si vede dal risultato, questi comandi hanno tutti lo stesso grado di elasticità:

sinistra   centro   destra
sinistra............................centro________________destra

È importante osservare che i comandi per gli spazi elastici non funzionano se si trovano all'inizio o alla fine di una riga. Eventualmente, se è necessario, si può inserire uno spazio non interrompibile ('**~**') primo o dopo il comando:

```
\noindent ~\hrulefill destra\par
\noindent sinistra\dotfill ~\par
```

____destra
sinistra..

50.9.2 Spazi verticali

I comandi per l'inserimento di spazi verticali riprendono la stessa logica di quelli orizzontali (escludendo il riempimento con puntini o linee). In generale, gli spazi rigidi sono cumulabili.

Comando	Descrizione
`\vspace{`***lunghezza***`}`	Inserisce uno spazio verticale rigido. Se si trova all'inizio di una pagina, viene ignorato.
`\vspace*{`***lunghezza***`}`	Inserisce uno spazio verticale rigido. Se si trova all'inizio di una pagina, viene inserito regolarmente.
`\addvspace{`***lunghezza***`}`	Aggiunge uno spazio verticale rigido. Se nello stesso punto vengono usati più comandi '**`\addvspace`**', viene preso in considerazione solo quello con la lunghezza maggiore.
`\vfill`	Inserisce uno spazio verticale elastico, che in condizioni normali va a riempire tutto lo spazio disponibile.

L'esempio seguente riguarda l'inserzione di spazi rigidi attraverso i comandi '**`\vspace`**' e '**`\addvspace`**':

```
1 Bla bla bla...

\vspace{0.5cm}
2 Bla bla bla...

\vspace{0.5cm}
\vspace{0.5cm}
3 Bla bla bla...

\vspace{0.5cm}
\addvspace{0.5cm}
\addvspace{1.5cm}
4 Bla bla bla...
```

Si può osservare che i comandi '**`\vspace`**' si sommano tra loro, mentre '**`\addvspace`**' no, limitando ad aggiungere lo spazio più grande:

1 Bla bla bla...

2 Bla bla bla...

3 Bla bla bla...

4 Bla bla bla...

Lo spazio verticale elastico consente di allineare il testo verticalmente, come nell'esempio seguente, in cui si individua un blocco centrale e un blocco in fondo alla pagina:

```
1 Sopra, bla bla bla...

\vfill
2 Centro, bla bla bla...

\vfill
3 Sotto, bla bla bla...

\newpage
```

1 Sopra, bla bla bla...

2 Centro, bla bla bla...

3 Sotto, bla bla bla...

Per l'uniformità della veste grafica del testo, è importante standardizzare gli spazi verticali, attraverso dei comandi appositi, che cambiano a seconda dello stile generale:

Comando	Descrizione
`\smallskip`	spazio piccolo;
`\medskip`	spazio medio;
`\bigskip`	spazio grande.

L'entità dello spazi verticale introdotto da questi comandi dipende da altri comandi:

Comando	Descrizione
`\smallskipamount`	lunghezza piccola;
`\medskipamount`	lunghezza media;
`\bigskipamount`	lunghezza grande.

In pratica, i comandi per l'inserzione di spazi verticali standard corrispondono all'uso di '**`\vskip`**' con queste lunghezze uniformi:

Comando	Equivalenza	Esempio di altezza
\smallskip	\vspace{\smallskipamount}	3.0pt plus 1.0pt minus 1.0pt
\medskip	\vspace{\medskipamount}	6.0pt plus 2.0pt minus 2.0pt
\bigskip	\vspace{\bigskipamount}	12.0pt plus 4.0pt minus 4.0pt

L'esempio seguente mostra in pratica l'uso di questi comandi per l'inserzione di spazi verticali standard:

```
Bla bla bla...\bigskip

Lo spazio prima di questa riga è grande;\medskip

lo spazio prima di questa riga è medio;\smallskip

lo spazio prima di questa riga è piccolo;

lo spazio prima di questa riga è quello normale.
```


Bla bla bla...

Lo spazio prima di questa riga è grande;

lo spazio prima di questa riga è medio;

lo spazio prima di questa riga è piccolo;
lo spazio prima di questa riga è quello normale.

50.9.3 Elasticità controllabile

È possibile definire in modo semplice una lunghezza elastica da usare poi con comandi come '**`\hspace`**' e '**`\vspace`**':

Comando	Descrizione
`\stretch{`***n***`}`	restituisce una lunghezza elastica pari a ***n***.

Maggiore è il valore attribuito all'argomento di '**`\stretch`**', maggiore è di conseguenza l'elasticità. A titolo si esempio si può vedere come si trasforma questo comando:

Comando	Lunghezza equivalente
\stretch{1}	0.0ptplus1fill
\stretch{2}	0.0ptplus2fill
\stretch{2.5}	0.0ptplus2.5fill

50.9.4 Interlinea

La distanza tra le righe di un blocco di testo normale si può controllare modificando la lunghezza '**`\baselineskip`**'; tuttavia, con l'aiuto del pacchetto '**`setspace`**', si possono usare comandi che semplificano questo tipo di intervento:

Comando	Descrizione
`\usepackage{setspace}`	dichiara l'utilizzo del pacchetto necessario alla gestione dei comandi sottostanti;
`\singlespacing`	richiede un'interlinea normale;
`\onehalfspacing`	richiede un'interlinea maggiore;
`\doublespacing`	richiede un'interlinea doppia.

L'esempio seguente mostra l'uso e l'effetto di questi comandi:

```
…
\usepackage{setspace}
…
\begin{document}

\singlespacing
Questo paragrafo ha un'interlinea normale. Bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla...

\onehalfspacing
Questo paragrafo ha un'interlinea pari a una volta e mezza
quella normale. Bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla...

\doublespacing
Questo paragrafo ha un'interlinea doppia. Bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla...

\end{document}
```

Questo paragrafo ha un'interlinea normale. 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...

Questo paragrafo ha un'interlinea pari a una volta e mezza quella normale. Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla...

Questo paragrafo ha un'interlinea doppia. Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla...

Anche senza il pacchetto '**`setspace`**', si può usare nel preambolo il

comando seguente:

Comando	Descrizione
`\linespread{`*fattore*`}`	moltiplica la distanza tra le righe del fattore indicato.

Per esempio, così:

```
…
\linespread{2.5}
…
\begin{document}

La distanza tra le righe è di due volte e mezza quella
normale. Bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla...

\end{document}
```

La distanza tra le righe è di due volte e mezza quella normale. Bla bla bla

bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla

bla...

50.9.5 Scatole

«

LaTeX offre una serie di comandi per la realizzazione di scatole, bordate e non, alcuni dei quali sono già apparsi in altri capitoli. Oltre alla presenza del bordo, le scatole si distinguono per la capacità o meno di rimpaginano il testo al loro interno.

Comando	Comando	Descrizione
`\mbox{`*testo*`}`	`\fbox{`*testo*`}`	Inserisce il testo in una scatola larga quanto il testo stesso; nel secondo caso si ottiene una scatola bordata.
`\makebox[`*ampiezza*`]`↩ ↪`[c|l|r|s]{`*testo*`}`	`\framebox[`*ampiezza*`]`↩ ↪`[c|l|r|s]{`*testo*`}`	Inserisce il testo (di una sola riga) in una scatola; nel secondo caso si ottiene una scatola bordata.

Quelli appena mostrati sono i comandi più comuni per ottenere delle scatole, in cui il testo occupa sempre una riga. Si può intuire che i comandi '**`\makebox`**' e '**`\framebox`**', usati senza argomenti opzionali, sono equivalenti rispettivamente a '**`\mbox`**' e '**`\fbox`**'. Il secondo argomento opzionale di '**`\makebox`**' e '**`\framebox`**' è una lettera che indica l'allineamento orizzontale che deve avere il testo contenuto:

Lettera	Significato mnemonico	Descrizione
`c`	*center*	testo centrato orizzontalmente nella scatola;
`l`	*left*	testo allineato a sinistra;
`r`	*right*	testo allineato a destra;
`s`	*stretch*	testo allargato per tutta l'ampiezza della scatola.

L'esempio seguente dovrebbe chiarire l'utilizzo di questi comandi:

```
Bla bla bla \mbox{ciao ciao} bla bla bla \fbox{ciao ciao}
bla bla bla...

Bla bla bla \makebox{ciao ciao} bla bla bla \framebox{ciao
ciao} bla bla bla...

Bla bla bla \makebox[4cm]{ciao ciao} bla bla bla
\framebox[4cm]{ciao ciao} bla bla bla...

Bla bla bla \makebox[4cm][c]{ciao ciao} bla bla bla
\framebox[4cm][c]{ciao ciao} bla bla bla...

Bla bla bla \makebox[4cm][l]{ciao ciao} bla bla bla
\framebox[4cm][l]{ciao ciao} bla bla bla...

Bla bla bla \makebox[4cm][r]{ciao ciao} bla bla bla
```



```
\framebox[4cm][r]{ciao ciao} bla bla bla...

Bla bla bla \makebox[4cm][s]{ciao ciao} bla bla bla
\framebox[4cm][s]{ciao ciao} bla bla bla...
```

Bla bla bla ciao ciao bla bla bla ciao ciao bla bla bla...
Bla bla bla ciao ciao bla bla bla ciao ciao bla bla bla...
Bla bla bla ciao ciao bla bla bla ciao ciao bla bla bla...
Bla bla bla ciao ciao bla bla bla ciao ciao bla bla bla...
Bla bla bla ciao ciao bla bla bla ciao ciao bla bla bla...
Bla bla bla ciao ciao bla bla bla ciao ciao bla bla bla...
Bla bla bla ciao ciao bla bla bla ciao ciao bla bla bla...

Le scatole bordate con '**`\fbox`**' e '**`\framebox`**' sono controllate da due dimensioni, molto utili per gestirne l'estetica:

Comando	Descrizione
`\fboxrule=`*dimensione* `\setlength{\fboxrule}{`*dimensione*`}`	definisce lo spessore della linea usata per il bordo;
`\fboxsep=`*dimensione* `\setlength{\fboxsep}{`*dimensione*`}`	definisce la distanza tra la linea del bordo e il contenuto della scatola.

L'esempio seguente dovrebbe chiarire in che modo usare questi valori per controllare le scatole bordate:

```
\fbox{scatola bordata normale}
{\fboxrule=3mm\fbox{scatola con bordo più spesso}}
\fbox{scatola bordata normale}
{\fboxsep=3mm\fbox{scatola più spaziosa}}
\fbox{scatola bordata normale}
{\fboxrule=3mm\fboxsep=3mm\fbox{scatola spaziosa con bordo
spesso}}
```

Si può osservare che le scatole «speciali» sono state racchiuse tra parentesi graffe, per evitare che la modifica delle caratteristiche legate allo spessore della linea e alla distanza dal contenuto, possa coinvolgere il resto del testo. L'alternanza con scatole normali dimostra il funzionamento di questo sistema di contenimento:

scatola bordata normale scatola con bordo più spesso scatola bordata normale
scatola più spaziosa scatola bordata normale scatola spaziosa con bordo spesso

Per inserire in una scatola più di una riga di testo, si possono usare due tipi di istruzioni, ottenendo risultati leggermente differenti:

Comando	Descrizione
`\parbox[c|t|b][`*altezza*`]`↩ ↪`[c|t|b|s]{`*larghezza*`}{`*testo*`}`	definisce una scatola, senza bordo, per contenere del testo impaginato nell'ambito della larghezza stabilita;

`\begin{minipage}[c|t|b]{`*larghezza*`}`
testo
`\end{minipage}`
	definisce una scatola, senza bordo, per contenere del testo impaginato nell'ambito della larghezza stabilita.

La differenza più importante tra questi due tipi di scatole sta nel fatto che '**`\parbox`**' consente l'inserimento di testo non troppo complesso, mentre l'ambiente '**`minipage`**' è letteralmente una piccola pagina, in cui anche le note a piè pagina rimangono al suo interno.

In entrambi i casi, il primo argomento opzionale indica la collocazione della scatola, in senso verticale, rispetto alla base della riga in cui ci si trova; pertanto:

Lettera	Significato mnemonico	Descrizione
`c`	*center*	richiede una scatola centrata verticalmente rispetto alla riga;
`t`	*top*	richiede una scatola che parte dalla riga e si espande in basso;
`b`	*bottom*	richiede una scatola che parte dalla riga e si espande in alto.

L'esempio seguente consente di confrontare il comportamento dei due tipi di scatole per ciò che riguarda l'allineamento di queste rispetto alla riga. Per facilitare il paragone visivo, queste scatole vengono bordate attraverso il comando '**`\fbox`**':

```
Bla bla bla bla bla...

Bla bla \fbox{\parbox{2cm}{Questa frase è in una scatola}}
bla \fbox{\parbox[c]{2cm}{Questa frase è in una scatola}}
bla \fbox{\parbox[t]{2cm}{Questa frase è in una scatola}}
bla \fbox{\parbox[b]{2cm}{Questa frase è in una scatola}}
bla bla.

Bla bla bla bla bla...

Bla bla \fbox{\begin{minipage}{2cm}Questa frase è in una
scatola\end{minipage}} bla
\fbox{\begin{minipage}[c]{2cm}Questa frase è in una
scatola\end{minipage}} bla
\fbox{\begin{minipage}[t]{2cm}Questa frase è in una
scatola\end{minipage}} bla
\fbox{\begin{minipage}[b]{2cm}Questa frase è in una
scatola\end{minipage}} bla bla.

Bla bla bla bla bla...
```

Come si vede dal risultato della composizione, per il momento non si notano differenze:

Bla bla bla bla bla...

Bla bla | Questa frase è in una scatola | bla | Questa frase è in una scatola | bla | Questa frase è in una scatola | bla | Questa frase è in una scatola

bla bla.

Bla bla bla bla bla...

Bla bla | Questa frase è in una scatola | bla | Questa frase è in una scatola | bla | Questa frase è in una scatola | bla | Questa frase è in una scatola

bla bla.

Bla bla bla bla bla...

Dalla sintassi mostrata a proposito di '**`\parbox`**', si nota la possibilità di indicare l'altezza della scatola, assieme a un terzo argomento opzionale che serve a collocare il testo nella scatola:

Lettera	Significato mnemonico	Descrizione
`c`	*center*	testo centrato verticalmente nella scatola;
`t`	*top*	testo in alto;
`b`	*bottom*	testo in basso;
`s`	*stretch*	testo allungato per tutta l'altezza della scatola.

L'esempio seguente riguarda esclusivamente '**`\parbox`**', per mostrare l'uso dei due ulteriori argomenti opzionali:

```
Bla bla bla bla bla...

Bla bla bla \fbox{\parbox[c][2cm]{2cm}{Questa frase è in una
scatola}}
\fbox{\parbox[c][2cm][c]{2cm}{Questa frase è in una scatola}}
\fbox{\parbox[c][2cm][t]{2cm}{Questa frase è in una scatola}}
\fbox{\parbox[c][2cm][b]{2cm}{Questa frase è in una scatola}}
\fbox{\parbox[c][2cm][s]{2cm}{Questa frase è in una scatola}}

Bla bla bla bla bla...
```

Si può osservare che nell'ultima scatola, l'opzione di allungamento verticale del testo non funziona, perché il testo non possiede spazi

elastici:

Bla bla bla bla bla...

Bla bla bla | Questa frase è in una scatola | Questa frase è in una scatola | Questa frase è in una scatola | Questa frase è in una scatola

Questa frase è in una scatola

Bla bla bla bla bla...

50.9.6 Testo da salvare e da recuperare

LaTeX ha una capacità limitata di memorizzare del testo in modo da poterlo inserire successivamente nella composizione:

Comando	Descrizione
`\begin{lrbox}{`***n***`}`*testo*`\end{lrbox}`	accumula nel registro ***n*** il testo indicato;
`\usebox{`***n***`}`	inserisce nella composizione il testo corrispondente al registro ***n***, in una scatola orizzontale.

La spiegazione dei due modelli sintattici dovrebbe essere sufficiente a comprendere il funzionamento della cosa; tuttavia è importante sottolineare che il testo accumulato può essere composto da una sola riga e viene poi rappresentato come se fosse inserito in una scatola '**`\mbox`**'. Segue un esempio:

```
\begin{lrbox}{1}
ciao a tutti
\end{lrbox}

\begin{lrbox}{2}
ma che bello
\end{lrbox}

Bla bla bla \usebox{2} bla bla bla \usebox{1} bla bla bla
\usebox{2} bla bla bla \usebox{1} bla bla bla...
```

Bla bla bla ma che bello bla bla bla ciao a tutti bla bla bla ma che bello bla bla bla ciao a tutti bla bla bla...

50.9.7 Linee

Con LaTeX, si ottengono linee orizzontali o verticali con il comando '**`\rule`**'. Per la precisione, si disegnano dei rettangoli neri di ampiezza e altezza stabiliti; pertanto, per ottenere una linea orizzontale si utilizza un rettangolo molto largo e molto basso, mentre per una linea verticale si usa un rettangolo molto stretto. Questi rettangoli sono collocati a loro volta nella parte superiore di una scatola, che in condizioni normali è alta quanto basta per contenere l'oggetto.

```
\rule{larghezza}{altezza}
```

```
\rule[spazio_verticale_aggiuntivo]{larghezza}{altezza}
```

L'esempio seguente riguarda la realizzazione di alcune linee, verticali e orizzontali, senza spazi verticali aggiuntivi:

```
Bla bla bla...

Bla bla bla \rule{0.5mm}{1cm} \rule{0.6mm}{1cm}
\rule{0.7mm}{1cm} \rule{0.8mm}{1cm} \rule{0.9mm}{1cm}
\rule{1mm}{1cm} bla bla bla...

\rule{\textwidth}{0.5mm}

\rule{\textwidth}{0.6mm}

\rule{\textwidth}{0.7mm}

\rule{\textwidth}{0.8mm}

\rule{\textwidth}{0.9mm}
```

```
\rule{\textwidth}{1mm}

Bla bla bla...
```

Bla bla bla...

Bla bla bla bla bla bla...

Bla bla bla...

L’esempio seguente è una variante con l’aggiunta di spazi verticali aggiuntivi:

```
Bla bla bla...

Bla bla bla \rule[1mm]{0.5mm}{1cm} \rule[2mm]{0.6mm}{1cm}
\rule[3mm]{0.7mm}{1cm} \rule[4mm]{0.8mm}{1cm}
\rule[5mm]{0.9mm}{1cm} \rule[6mm]{1mm}{1cm} bla bla bla...

\rule[1mm]{\textwidth}{0.5mm}

\rule[2mm]{\textwidth}{0.6mm}

\rule[3mm]{\textwidth}{0.7mm}

\rule[4mm]{\textwidth}{0.8mm}

\rule[5mm]{\textwidth}{0.9mm}

\rule[6mm]{\textwidth}{1mm}

Bla bla bla...
```

Bla bla bla...

Bla bla bla bla bla bla...

Bla bla bla...

50.10 Riferimenti incrociati

LaTeX gestisce diversi tipi di riferimenti incrociati, di cui il più importante è l’indice generale, ottenuto attraverso il comando ‘**`\tableofcontents`**’. La gestione dei riferimenti incrociati avviene attraverso la creazione di file temporanei, adibiti ad accumulare le informazioni, da riutilizzare attraverso rielaborazioni successive. In pratica, è proprio la presenza di questi riferimenti incrociati che obbliga a ripetere il procedimento di composizione più volte.

50.10.1 Indice generale, indice delle tabelle e delle figure

La creazione dell’indice generale, dell’indice delle tabelle e dell’indice delle figure, parte dall’accumulo delle informazioni sulla collocazione di queste voci. Per la precisione, i comandi che definiscono la suddivisione del documento annotano il titolo, il numero della sezione e la pagina in un file con estensione ‘`.toc`’ (*Table of contents*); i comandi ‘**`\caption`**’ inseriti nell’involucro fluttuante di una tabella o di una figura, annotano il numero e la pagina in cui si trovano in file con estensione ‘`.lot`’ (*List of tables*) e ‘`.lof`’ (*List of figures*).

Nella rielaborazione successiva (quando il comando di composizione viene ripetuto), questi file possono essere incorporati attraverso i comandi ‘**`\tableofcontents`**’, ‘**`\listoftables`**’ e ‘**`\listoffigures`**’, per ottenere rispettivamente l’indice generale, l’indice delle tabelle e delle figure.

Dal momento che l’inserzione modifica la lunghezza del testo, a meno che gli indici siano collocati alla fine del documento, si rende necessaria almeno una terza rielaborazione.

Comando	Descrizione
`\tableofcontents`	inserisce nella composizione l’indice generale, ottenuto dal file ‘`.toc`’, generato dalla composizione precedente;
`\listoftables`	inserisce nella composizione l’indice delle tabelle, ottenuto dal file ‘`.lot`’, generato dalla composizione precedente;
`\listoffigures`	inserisce nella composizione l’indice delle figure, ottenuto dal file ‘`.lof`’, generato dalla composizione precedente.

L’inserimento di voci in questi elenchi può avvenire anche in modo manuale, attraverso due comandi appositi:

Comando	Descrizione
`\addtocontents{toc｜lot｜lof}{`***testo***`}`	aggiunge il testo indicato nell’ultimo argomento, nell’indice corrispondente alla sigla del primo argomento (indice generale, indice delle tabelle o indice delle figure);
`\addcontentsline↩` `↪{toc｜lot｜lof}{`***livello***`}{`***testo***`}`	aggiunge la voce nell’indice corrispondente alla sigla del primo argomento.

Il funzionamento del comando ‘**`\addtocontents`**’ è relativamente semplice, nel senso che ci si limita ad aggiungere del testo nell’indice prescelto:

```
\tableofcontents

\section{Prima}

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...

\addtocontents{toc}{Ciao a tutti\par}

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...

\section{Seconda}

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...
```

Si può osservare che nel testo inserito è stata aggiunta la richiesta esplicita di concludere il paragrafo; ciò si rende necessario per evitare problemi imprevisti nel momento dell’incorporazione del file ‘`.toc`’.

Indice

1 Prima

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 bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla...

2 Seconda

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...

L’uso di ‘**`\addcontentsline`**’ è più complesso, perché richiede anche l’indicazione del «livello» della voce. Se si tratta dell’indice generale, il livello è rappresentato dalle parole chiave da ‘**`part`**’ a ‘**`subparagraph`**’, mentre negli altri casi è ammesso l’uso di ‘**`table`**’ o ‘**`figure`**’. Così si intende che la voce inserita sia

da trattare allo stesso modo del livello corrispondente, per ciò che riguarda la composizione. Nell'esempio che segue, la voce si inserisce al livello di una sottosezione:

```
\tableofcontents

\section{Prima}

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...

\addcontentsline{toc}{subsection}{Ciao a tutti}

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...

\section{Seconda}

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...
```

Come si vede, in questo caso non c'è più bisogno di concludere il paragrafo nel testo, cosa che al contrario genererebbe qui solo un errore:

Indice

1 Prima

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 bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla...

2 Seconda

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...

Volendo, si può spiare il contenuto del file '`.toc`' generato, per scoprire come si può aggiungere una cifra davanti alla voce inserita manualmente:

```
\contentsline {section}{\numberline {1}Prima}{1}
\contentsline {section}{Ciao a tutti}{1}
\contentsline {section}{\numberline {2}Seconda}{1}
```

Intuitivamente si comprende l'uso del comando '**`\numberline`**':

```
\tableofcontents

\section{Prima}

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...

\addcontentsline{toc}{subsection}{\numberline{1.1}Ciao a
tutti}

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...

\section{Seconda}

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...
```


Indice

1 Prima

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 bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla...

2 Seconda

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...

50.10.2 Riferimenti liberi

LaTeX mette a disposizione pochi comandi per la creazione di riferimenti incrociati all'interno del testo. Attraverso il comando '**`\label`**' è possibile definire un'etichetta alla quale si può fare riferimento con i comandi '**`\ref`**' o '**`\pageref`**':

Comando	Descrizione
`\label{`*stringa_identificativa*`}`	mette un punto di riferimento invisibile, identificato dalla stringa contenuta nell'argomento;
`\ref{`*stringa_identificativa*`}`	inserisce nella composizione il numero corrispondente alla stringa di identificazione indicata, secondo il contesto;
`\pageref{`*stringa_identificativa*`}`	inserisce nella composizione il numero della pagina corrispondente alla stringa di identificazione indicata.

La posizione in cui viene collocato il comando '**`\label`**' è importante, in quanto '**`\ref`**' si comporta diversamente a seconda del contesto. Se l'etichetta viene dichiarata all'interno di testo normale, il riferimento generico a questa restituisce un numero, più o meno articolato, che indica la sezione o il capitolo in cui si trova; se invece l'etichetta viene dichiarata all'interno di un ambiente numerato, come una tabella, una figura, un'equazione o altro, il riferimento a questa genera il numero corrispondente a tale elemento.

A titolo di esempio, viene mostrato come si potrebbe procedere per la costruzione manuale di un indice generale, senza l'ausilio del comando '**`\tableofcontents`**':

```
\section*{Indice}

\noindent\ref{sec:Prima} Prima, pag. \pageref{sec:Prima}

\noindent\ref{sec:Seconda} Seconda, pag.
\pageref{sec:Seconda}

\section{Prima} \label{sec:Prima}

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...

\section{Seconda} \label{sec:Seconda}

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...
```

Indice

1 Prima

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...

2 Seconda

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...

È bene ribadire che LaTeX inserisce implicitamente le etichette generate con il comando '**\label**' in un contesto particolare, per cui il risultato del comando '**\ref**' cambia di conseguenza. Tuttavia, il numero che viene mostrato da '**\ref**' non è accompagnato dal contesto in fase di composizione, pertanto è compito dell'autore scrivere di cosa si tratta (capitolo, sezione, tabella, figura o altro). In questo senso, diventa importante ricordare a cosa si riferisce una certa stringa di identificazione, pertanto, la stessa documentazione originale di LaTeX propone di usare un prefisso diverso in base al contesto:

Comando	Descrizione
`\label{cha:`*stringa_successiva*`}`	etichetta di identificazione di un capitolo;
`\label{sec:`*stringa_successiva*`}`	etichetta di identificazione di una sezione di livello inferiore al capitolo;
`\label{tab:`*stringa_successiva*`}`	etichetta di identificazione di una tabella;
`\label{fig:`*stringa_successiva*`}`	etichetta di identificazione di una figura;
`\label{eq:`*stringa_successiva*`}`	etichetta di identificazione di un'equazione.

Le stringhe di identificazione generate dal comando '**\label**', sono annotate in un file con estensione '`.aux`' (assieme ad altre informazioni estranee). Quando LaTeX viene riavviato in una fase di composizione successiva, carica in memoria i dati contenuti nel file '`.aux`' e lo utilizza per risolvere i riferimenti con i comandi '**\ref**' e '**\pageref**'.

50.10.3 Indice analitico

La realizzazione di un indice analitico, ovvero un indice che raccoglie una serie ordinata di voci che possono essere inserite in varie parti del documento, è più complessa rispetto a quella dell'indice generale. In special modo, in questo caso è necessario provvedere a riordinare e riorganizzare i dati nel modo corretto. I comandi coinvolti per la realizzazione dell'indice analitico, sono riassunti brevemente nell'elenco seguente:

Comando	Descrizione
`\usepackage{makeidx}`	si usa questo comando nel preambolo per importare il codice necessario alla gestione dell'indice analitico;
`\makeindex`	si usa questo comando, sempre nel preambolo, per attivare la gestione dell'indice analitico;
`\index{`*voce*`}`	inserisce la voce indicata nel file '`.idx`', dal quale si ottiene poi l'indice analitico;
`\printindex`	inserisce l'indice analitico contenuto nel file '`.ind`', ottenuto attraverso un'elaborazione esterna a LaTeX.

Come si vede, è necessario utilizzare un pacchetto esterno, denominato '**makeidx**', quindi si deve attivare la costruzione dell'indice, con il comando '**\makeindex**'. Pertanto, si inseriscono le istruzioni seguenti nel preambolo:

```
\usepackage{makeidx}
\makeindex
```

Lo scopo particolare di '**\makeindex**' è quello di aprire un file con estensione '`.idx`', in cui il comando '**\index**' può poi annotare le sue voci. Si osservi l'esempio seguente:

```
…
\usepackage{makeidx}
\makeindex
…
\begin{document}

\section{Prima} \index{resistenza}\index{induttanza}

Qui si parla di resistenze e di induttanze, 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...

\section{Seconda} \index{capacità}

Qui si parla di capacità e simili, 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...

\end{document}
```

Nella composizione finale, i comandi '**\index**' non producono alcun risultato e si comportano in modo simile al comando '**\label**':

1 Prima

Qui si parla di resistenze e di induttanze, 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...

2 Seconda

Qui si parla di capacità e ancora di resistenza, 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...

Dopo l'elaborazione normale con LaTeX si ottiene un file con estensione '`.idx`', con il contenuto seguente:

```
\indexentry{resistenza}{1}
\indexentry{induttanza}{1}
\indexentry{capacit\‘a}{1}
\indexentry{resistenza}{1}
```

Si può osservare che le voci sono state inserite nella sequenza in cui appaiono nel sorgente LaTeX, senza tenere conto di eventuali ripetizioni. Ma LaTeX non è nelle condizioni di usare questo file direttamente nell'elaborazione successiva; prima occorre riordinare e riorganizzare le voci attraverso il programma '**makeindex**':

```
makeindex nome_file_senza_estensione
```

Per esempio, se si sta facendo la composizione del file '`prova.tex`', è sufficiente il comando:

```
$ makeindex prova [Invio]
```

In questo modo, '**makeindex**' cerca di leggere il file '`prova.idx`' (quello generato dall'elaborazione di LaTeX) e produce il file '`prova.ind`', che, seguendo l'esempio, dovrebbe contenere quanto segue:

```
\begin{theindex}

  \item capacit\‘a, 1

  \indexspace

  \item induttanza, 1

  \indexspace

  \item resistenza, 1

\end{theindex}
```

Come si può vedere, la voce '**resistenza**' contiene un riferimento unico, dal momento che appare due volte, ma nella stessa pagina finale.[3]

Se il sorgente LaTeX contiene il comando '**\printindex**', in quel

punto viene inserito il contenuto del file con estensione '.ind', a partire da una pagina nuova, con l'aggiunta di un titolo adeguato:

```
…
\usepackage{makeidx}
\makeindex
…
\begin{document}

\section{Prima} \index{resistenza}\index{induttanza}

Qui si parla di resistenze e di induttanze, 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...

\section{Seconda} \index{capacità}

Qui si parla di capacità e simili, 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...

\printindex

\end{document}
```

1 Prima

Qui si parla di resistenze e di induttanze, 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...

2 Seconda

Qui si parla di capacità e ancora di resistenza, 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...

Indice analitico

capacità, 1

induttanza, 1

resistenza, 1

1

È da osservare che la voce indicata come argomento del comando '**\index**' può essere più articolata, in base alle capacità elaborative di '**makeindex**'. In pratica, si utilizzano dei simboli per separare la voce in più parti. L'elenco seguente dovrebbe aiutare a comprendere il meccanismo:

Comando	Descrizione
\index{*voce*}	questa è la situazione normale, in cui si inserisce la voce indicata, così come inserita, nell'indice;
\index{*voce_principale* ! *voce_secondaria*}	il punto esclamativo permette di separare una voce principale da una parte secondaria, che poi può essere mostrata con un leggero rientro rispetto alla prima parte;
\index{*voce_principale@voce_abbellita*}	la chiocciola separa una prima parte, da prendere in considerazione ai fini dell'ordinamento, dalla seconda parte che invece viene mostrata nell'indice, consentendo l'uso di comandi che modificano l'aspetto del carattere;
\index{*voce*↩ ↪| *comandi_per_il_numero_della_pagina*}	la barra verticale separa la voce dell'indice dai comandi per controllare l'aspetto del numero della pagina (comandi da usare senza la barra obliqua inversa iniziale).

L'esempio seguente è una variante di quanto già visto, per mostrare

l'uso del comando '**\index**' con questi simboli particolari:

```
\section{Prima}
\index{resistenza}\index{resistenza!induttiva|textit}

Qui si parla di resistenze e di induttanze, 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...

\section{Seconda} \index{resistenza!capacitiva|textit}
\index{resistenza@\texttt{resistenza}}

Qui si parla di capacità e ancora di resistenza, 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...
```

Ecco come si presenta l'indice; si può osservare che la voce '**resistenza**' appare due volte, perché il tipo di carattere è stato modificato:

Indice analitico

resistenza, 1
  capacitiva, *1*
  induttiva, *1*
`resistenza`, 1

1

50.10.4 Bibliografie

«

LaTeX offre una gestione molto semplice delle bibliografie attraverso l'uso dell'ambiente '**thebibliography**'. I comandi coinvolti sono descritti brevemente nell'elenco seguente:

Comando	Descrizione
\begin{thebibliography}↩ ↪{*stringa_di_esempio*}...↩ ↪\end{thebibliography}	dichiara l'ambiente che contiene le informazioni bibliografiche;
\bibitem[*sigla*]↩ ↪{*stringa_identificativa_della_voce*}	all'interno dell'ambiente '**thebibliography**' dichiara l'inizio di una voce bibliografica;
\cite[*testo_aggiuntivo*↩ ↪{*stringa_identificativa_della_voce*}}	si usa al di fuori dell'ambiente '**thebibliography**', per richiamare il riferimento a una voce bibliografica.

In pratica, l'ambiente '**thebibliography**' serve a raccogliere le voci bibliografiche, ognuna dichiarata attraverso il comando '**\bibitem**'. Nel risultato della composizione, le voci vengono precedute da una sigla, come indicato dal comando '**\bibitem**', oppure da un numero gestito in modo automatico; pertanto, per garantire un allineamento corretto, la dichiarazione dell'ambiente richiede un argomento composto da una stringa di esempio, che rappresenti l'ampiezza massima di questa sigla, sia gestita manualmente, sia automatica.

Al di fuori dell'ambiente '**thebibliography**', di solito prima di questo, si usa il comando '**\cite**' per fare riferimento a una voce bibliografica, attraverso la stringa identificativa che usa lo stesso comando '**\bibitem**'. Se viene usato l'argomento opzionale di '**\cite**', si ottiene l'aggiunta di quella stringa dopo la sigla usata per identificarlo.

Per cominciare conviene mostrare un esempio molto semplice, in cui non si usano argomenti opzionali:

```
\section{Resistenze e condensatori}

Qui si parla di resistenze \cite{res} e di condensatori
\cite{con}, bla bla bla bla bla bla bla bla bla...

\begin{thebibliography}{99}
\bibitem{res} Tizio Tizi, \textit{Resistenze elettriche},
2002
\bibitem{con} Caio Cai, \textit{Condensatori e correnti
```

```
alternate}, 2001
\end{thebibliography}
```

Come si vede, l'argomento della dichiarazione dell'ambiente '**thebibliography**' contiene la stringa '**99**', perché si sa che la sigla usata per distinguere le voci non può essere più lunga di un numero di due cifre:

1 Resistenze e condensatori

Qui si parla di resistenze [1] e di condensatori [2], bla bla bla bla bla bla bla bla bla...

Riferimenti bibliografici

[1] Tizio Tizi, *Resistenze elettriche*, 2002

[2] Caio Cai, *Condensatori e correnti alternate*, 2001

Volendo aggiungere un riferimento a una certa pagina del primo libro citato, basta sfruttare l'argomento opzionale di '**\cite**':

```
Qui si parla di resistenze \cite[pag. 1]{res} e di
condensatori \cite{con}, bla bla bla bla bla bla bla bla
bla...
```

1 Resistenze e condensatori

Qui si parla di resistenze [1, pag. 1] e di condensatori [2], bla bla bla bla bla bla bla bla bla...

Riferimenti bibliografici

[1] Tizio Tizi, *Resistenze elettriche*, 2002

[2] Caio Cai, *Condensatori e correnti alternate*, 2001

Infine, per usare delle sigle più significative, si può sfruttare l'argomento opzionale del comando '**\bibitem**', ma in questo caso occorre intervenire anche nella dichiarazione dell'ampiezza massima di queste stringhe:

```
\begin{thebibliography}{ZZZZZ}
\bibitem[TIZ]{res} Tizio Tizi, \textit{Resistenze
elettriche}, 2002
\bibitem[COND]{con} Caio Cai, \textit{Condensatori e
correnti alternate}, 2001
\end{thebibliography}
```

1 Resistenze e condensatori

Qui si parla di resistenze [TIZ, pag. 1] e di condensatori [COND], bla bla bla bla bla bla bla bla bla...

Riferimenti bibliografici

[TIZ] Tizio Tizi, *Resistenze elettriche*, 2002

[COND] Caio Cai, *Condensatori e correnti alternate*, 2001

Come si intuisce, l'uso di riferimenti bibliografici di questo tipo richiede delle rielaborazioni successive, anche se in questo caso non intervengono programmi esterni.

Eventualmente, se questo tipo di gestione bibliografica si mostra insufficiente, è disponibile BibTeX, che consente di gestire una bibliografia molto dettagliata in un file esterno, dove solo le voci richieste espressamente vengono poi mostrate nella composizione finale.

50.11 Tabelle

LaTeX offre due ambienti per la costruzione di tabelle: '**tabbing**' e '**tabular**'. Il primo dei due riguarda la realizzazione di tabelle attraverso delle tabulazioni, mentre il secondo è invece più completo, anche se poi offre meno libertà.

50.11.1 Tabella fatta di tabulazioni

Attraverso l'ambiente '**tabbing**', si può allineare il testo in colonne, fissando gli stop di tabulazione attraverso scatole orizzontali di riferimento, che di solito si ottengono con del testo opportuno. Si osservi subito un esempio molto semplice:


```
\begin{tabbing}
marrone \= 10 \% \= quattro \kill

nero \> 0 \> zero \\
marrone \> 1 \> uno \\
rosso \> 2 \> due \\
arancio \> 3 \> tre \\
giallo \> 4 \> quattro \\
verde \> 5 \> cinque \\
blu \> 6 \> sei \\
viola \> 7 \> sette \\
grigio \> 8 \> otto \\
bianco \> 9 \> nove \\

argento \> 10 \% \\
oro \> 5 \% \\
\> 2 \% \\
\> 1 \% \\
\end{tabbing}
```

Si può già comprendere intuitivamente il significato delle istruzioni inserite nell'esempio. In particolare, si può osservare nel risultato della composizione che la prima riga non appare, ma serve solo per indicare alle righe successive la collocazione degli stop di tabulazione; infatti, si può vedere che la prima riga contiene l'esempio del testo più lungo utilizzato in ogni colonna:

nero	0	zero
marrone	1	uno
rosso	2	due
arancio	3	tre
giallo	4	quattro
verde	5	cinque
blu	6	sei
viola	7	sette
grigio	8	otto
bianco	9	nove
argento	10 %	
oro	5 %	
	2 %	
	1 %	

La sintassi per l'utilizzo di questo ambiente '**tabbing**' è rappresentata in modo semplificato dal modello seguente:

```
\begin{tabbing}
testo \= testo [\= testo]... \kill
testo \> testo [\> testo]... \\
[testo \> testo [\> testo]... \\]
...
\end{tabbing}
```

Il modello non esaurisce tutte le possibilità, ma rende l'idea dell'utilizzo più frequente: all'inizio si dichiara la distanza degli stop di tabulazione con una riga di esempio, che poi non viene prodotta nella composizione. Tra le alternative possibili, c'è da considerare la possibilità di definire gli stop di tabulazione in una riga che poi viene comunque emessa anche nella composizione (in tal caso si sostituisce '**\kill**' con '****'); inoltre, è possibile modificare gli stop di tabulazione con un'altra riga contenente comandi '**\=**'. Segue la descrizione dei comandi più importanti che riguardano l'ambiente '**tabular**':

Comando	Descrizione
`\=`	fissa uno stop di tabulazione nella posizione in cui si trova;
`\>`	salta allo stop di tabulazione successivo;
`\kill`	prende in considerazione i comandi che fissano gli stop di tabulazione, ma non mostra la riga corrispondente nella composizione finale;
`\pushtabs` `\poptabs`	salva e ripristina le posizioni degli stop di tabulazione, quando si vuole avere una suddivisione diversa in un ambito temporaneo della tabella.

In realtà, i comandi specifici dell'ambiente '**tabbing**' sono molti di più; in particolare, la sovrapposizione di accenti alle lettere funziona

in modo diverso dal solito. Onde evitare di creare confusione per questo motivo, si preferisce mostrare un modo alternativo di usare l'ambiente '**tabbing**', allo scopo di raggirare questo inconveniente:

```
\begin{tabbing}
testo [\= testo]… \kill
\parbox{\textwidth}{testo} [\> \parbox{\textwidth}{testo}]… \\
[\parbox{\textwidth}{testo} [\> \parbox{\textwidth}{testo}]
… \\]
…
\end{tabbing}
```

In pratica, si inserisce il testo di ogni cella dentro l'ambiente definito dal comando '**\parbox**', che ripristina le condizioni di funzionamento normale.

Per definire in modo preciso gli stop di tabulazione, ci si può avvalere del comando '**\hspace**', il cui scopo è quello di produrre uno spazio orizzontale ben definito:

```
\begin{tabbing}
\hspace*{largh_col} [\= \hspace*{largh_col}]… \kill
\parbox{\textwidth}{testo} ↩
↪[\> \parbox{\textwidth}{testo}]… \\
[\parbox{\textwidth}{testo} ↩
↪[\> \parbox{\textwidth}{testo}]… \\]
…
\end{tabbing}
```

L'asterisco che appare nel comando '**\hspace**' è un'opzione necessaria a evitare che lo spazio finale venga eliminato (diversamente non si otterrebbe l'effetto desiderato). Viene ripreso l'esempio iniziale, modificato con le tecniche appena descritte, aggiungendo una colonna ulteriore per dei commenti:

```
\begin{tabbing}
\hspace*{3cm} \= \hspace*{2cm} \= \hspace*{3cm} \= \hspace*{4cm} \kill

\parbox{\textwidth}{nero}    \> \parbox{\textwidth}{0}
    \> \parbox{\textwidth}{zero} \\
\parbox{\textwidth}{marrone} \> \parbox{\textwidth}{1}
    \> \parbox{\textwidth}{uno} \\
\parbox{\textwidth}{rosso}   \> \parbox{\textwidth}{2}
    \> \parbox{\textwidth}{due} \\
\parbox{\textwidth}{arancio} \> \parbox{\textwidth}{3}
    \> \parbox{\textwidth}{tre} \\
\parbox{\textwidth}{giallo}  \> \parbox{\textwidth}{4}
    \> \parbox{\textwidth}{quattro} \\
\parbox{\textwidth}{verde}   \> \parbox{\textwidth}{5}
    \> \parbox{\textwidth}{cinque} \\
\parbox{\textwidth}{blu}     \> \parbox{\textwidth}{6}
    \> \parbox{\textwidth}{sei} \\
\parbox{\textwidth}{viola}   \> \parbox{\textwidth}{7}
    \> \parbox{\textwidth}{sette} \\
\parbox{\textwidth}{grigio}  \> \parbox{\textwidth}{8}
    \> \parbox{\textwidth}{otto} \\
\parbox{\textwidth}{bianco}  \> \parbox{\textwidth}{9}
    \> \parbox{\textwidth}{nove} \\

\parbox{\textwidth}{argento} \> \parbox{\textwidth}{10 \%}
  \> \> \parbox{\textwidth}{bla bla bla bla bla bla bla bla bla bla} \\
\parbox{\textwidth}{oro} \> \parbox{\textwidth}{5 \%} \\
                         \> \parbox{\textwidth}{2 \%} \\
                         \> \parbox{\textwidth}{1 \%} \\
\end{tabbing}
```

Per il momento, le tecniche mostrate non risolvono ancora il problema del traboccamento del testo da una colonna a quella successiva. A questo proposito, nell'esempio si può osservare che il commento nella quarta colonna, per la quale si prevede una larghezza di 4 cm, va oltre lo spazio a disposizione.

nero	0	zero	
marrone	1	uno	
rosso	2	due	
arancio	3	tre	
giallo	4	quattro	
verde	5	cinque	
blu	6	sei	
viola	7	sette	
grigio	8	otto	
bianco	9	nove	
argento	10 %		bla bla bla bla bla bla bla bla bla bla bla
oro	5 %		
	2 %		
	1 %		

Per ovviare a questo inconveniente, occorre controllare meglio il comando '**\parbox**', specificando esattamente la larghezza, nello stesso modo usato per la definizione degli stop di tabulazione:

```
\begin{tabbing}
\hspace*{3cm} \= \hspace*{2cm} \= \hspace*{3cm}
              \= \hspace*{4cm} \kill

\parbox{3cm}{nero}    \> \parbox{2cm}{0}
    \> \parbox{3cm}{zero} \\
\parbox{3cm}{marrone} \> \parbox{2cm}{1}
    \> \parbox{3cm}{uno} \\
\parbox{3cm}{rosso}   \> \parbox{2cm}{2}
    \> \parbox{3cm}{due} \\
\parbox{3cm}{arancio} \> \parbox{2cm}{3}
    \> \parbox{3cm}{tre} \\
\parbox{3cm}{giallo}  \> \parbox{2cm}{4}
    \> \parbox{3cm}{quattro} \\
\parbox{3cm}{verde}   \> \parbox{2cm}{5}
    \> \parbox{3cm}{cinque} \\
\parbox{3cm}{blu}     \> \parbox{2cm}{6}
    \> \parbox{3cm}{sei} \\
\parbox{3cm}{viola}   \> \parbox{2cm}{7}
    \> \parbox{3cm}{sette} \\
\parbox{3cm}{grigio}  \> \parbox{2cm}{8}
    \> \parbox{3cm}{otto} \\
\parbox{3cm}{bianco}  \> \parbox{2cm}{9}
    \> \parbox{3cm}{nove} \\

\parbox{3cm}{argento} \> \parbox{2cm}{10 \%}
    \> \> \parbox{4cm}{bla bla bla bla bla bla bla
                       bla bla bla bla} \\
\parbox{3cm}{oro} \> \parbox{2cm}{5 \%} \\
                     \> \parbox{2cm}{2 \%} \\
                     \> \parbox{2cm}{1 \%} \\
\end{tabbing}
```

In questo modo, finalmente, il testo che eccede lo spazio a disposizione viene suddiviso automaticamente in più righe; inoltre, è possibile controllare la posizione verticale del testo nella cella, attraverso le opzioni di '**\parbox**'.

nero	0	zero	
marrone	1	uno	
rosso	2	due	
arancio	3	tre	
giallo	4	quattro	
verde	5	cinque	
blu	6	sei	
viola	7	sette	
grigio	8	otto	
bianco	9	nove	
argento	10 %		bla bla bla bla bla bla bla bla bla bla bla bla
oro	5 %		
	2 %		
	1 %		

L'ultimo problema che vale la pena di prendere in considerazione in questo tipo di ambiente, è la definizione di tabulazioni relative all'ampiezza del testo. Ciò si ottiene anteponendo un numero davanti al comando che restituisce già questo valore. Per esempio, '**0.5\textwidth**' rappresenta esattamente la metà dell'ampiezza orizzontale a disposizione. Viene riproposto l'esempio consueto con l'indicazione di ampiezze relative:

```
\begin{tabbing}
\hspace*{0.2\textwidth} \= \hspace*{0.1\textwidth}
    \= \hspace*{0.2\textwidth}
    \= \hspace*{0.8\textwidth} \kill

\parbox{0.2\textwidth}{nero}    \> \parbox{0.1\textwidth}{0}
```

```
    \> \parbox{0.2\textwidth}{zero} \\
\parbox{0.2\textwidth}{marrone} \> \parbox{0.1\textwidth}{1}
    \> \parbox{0.2\textwidth}{uno} \\
\parbox{0.2\textwidth}{rosso}   \> \parbox{0.1\textwidth}{2}
    \> \parbox{0.2\textwidth}{due} \\
\parbox{0.2\textwidth}{arancio} \> \parbox{0.1\textwidth}{3}
    \> \parbox{0.2\textwidth}{tre} \\
\parbox{0.2\textwidth}{giallo}  \> \parbox{0.1\textwidth}{4}
    \> \parbox{0.2\textwidth}{quattro} \\
\parbox{0.2\textwidth}{verde}   \> \parbox{0.1\textwidth}{5}
    \> \parbox{0.2\textwidth}{cinque} \\
\parbox{0.2\textwidth}{blu}     \> \parbox{0.1\textwidth}{6}
    \> \parbox{0.2\textwidth}{sei} \\
\parbox{0.2\textwidth}{viola}   \> \parbox{0.1\textwidth}{7}
    \> \parbox{0.2\textwidth}{sette} \\
\parbox{0.2\textwidth}{grigio}  \> \parbox{0.1\textwidth}{8}
    \> \parbox{0.2\textwidth}{otto} \\
\parbox{0.2\textwidth}{bianco}  \> \parbox{0.1\textwidth}{9}
    \> \parbox{0.2\textwidth}{nove} \\

\parbox{0.2\textwidth}{argento}
    \> \parbox{0.1\textwidth}{10 \%}
    \> \> \parbox{0.3\textwidth}{bla bla bla bla bla bla bla
                                 bla bla bla bla} \\
\parbox{0.2\textwidth}{oro}
    \> \parbox{0.1\textwidth}{5 \%} \\
    \> \parbox{0.1\textwidth}{2 \%} \\
    \> \parbox{0.1\textwidth}{1 \%} \\
\end{tabbing}
```

nero	0	zero
marrone	1	uno
rosso	2	due
arancio	3	tre
giallo	4	quattro
verde	5	cinque
blu	6	sei
viola	7	sette
grigio	8	otto
bianco	9	nove
argento	10 %	bla bla bla bla bla bla bla bla bla bla bla bla
oro	5 %	
	2 %	
	1 %	

Questo tipo di tabella, ottenuta attraverso degli stop di tabulazione, è fatta per poter essere separata anche in più pagine; inoltre, non si adatta facilmente all'inserimento di bordi attorno alle celle. Tuttavia, per la gestione dei bordi si preferisce di solito l'ambiente '**`tabular`**'.

50.11.2 Tabella fatta di colonne e righe

Anche se in pratica si tratta sempre della stessa cosa, l'ambiente '**`tabular`**' non fa riferimento a degli stop di tabulazione, ma a delle colonne, definite contestualmente alla dichiarazione iniziale dell'ambiente:

```
\begin{tabular}{definizione_delle_colonne}
testo & testo [& testo]... \\
[testo & testo [& testo]... \\]
...
\end{tabular}
```

Il modello appena mostrato è semplificato rispetto alle possibilità complessive dell'ambiente '**`tabular`**'. Come si può intuire, il simbolo '**`&`**' viene usato per identificare la fine di una colonna e l'inizio di quella successiva; inoltre, come già avveniva per l'ambiente '**`tabbing`**', si usa il comando '**`\\`**' per dichiarare la fine di una riga della tabella.

La definizione delle colonne avviene attraverso una stringa particolare che può essere anche molto articolata. Per cominciare in modo semplice, si può usare una lettera per ogni colonna, dove la lettera ha un significato specifico:

Lettera	Descrizione
`l`	rappresenta una colonna il cui contenuto deve essere allineato a sinistra;

Lettera	Descrizione
`r`	rappresenta una colonna il cui contenuto deve essere allineato a destra;
`c`	rappresenta una colonna il cui contenuto deve essere allineato al centro.

Inoltre, esiste la possibilità di segnalare l'inserzione di una linea verticale alle estremità oppure tra una colonna e l'altra:

Simbolo	Descrizione
`|`	rappresenta una separazione attraverso una linea verticale.

Per esempio, la tabella che viene dichiarata nel modo seguente, indica la presenza di tre colonne, dove la prima e la terza ha il testo allineato a sinistra, mentre quella centrale ha il testo allineato a destra; inoltre, definisce un bordo verticale sinistro, un bordo destro e un bordo tra la prima e la seconda colonna.

```
\begin{tabular}{|l|rl|}
```

Viene riproposto un esempio già mostrato a proposito dell'ambiente '**`tabbing`**', secondo questa definizione di colonne:

```
\begin{tabular}{|l|rl|}
nero            & 0     & zero          \\
marrone         & 1     & uno           \\
rosso           & 2     & due           \\
arancio         & 3     & tre           \\
giallo          & 4     & quattro       \\
verde           & 5     & cinque        \\
blu             & 6     & sei           \\
viola           & 7     & sette         \\
grigio          & 8     & otto          \\
bianco          & 9     & nove          \\
argento         & 10 \% &               \\
oro             & 5 \%  &               \\
                & 2 \%  &               \\
                & 1 \%  &               \\
\end{tabular}
```

nero	0	zero
marrone	1	uno
rosso	2	due
arancio	3	tre
giallo	4	quattro
verde	5	cinque
blu	6	sei
viola	7	sette
grigio	8	otto
bianco	9	nove
argento	10 %	
oro	5 %	
	2 %	
	1 %	

Come si vede, non è stata dichiarata l'ampiezza delle colonne, pertanto questa viene determinata automaticamente, prendendo in considerazione l'ampiezza della cella che contiene il testo più lungo, anche se questo può produrre una tabella più larga dello spazio a disposizione nel corpo del documento.

Sempre rimanendo a un livello realizzativo molto semplice, si possono inserire delle linee orizzontali, attraverso il comando '**`\hline`**', come nella variante seguente, in cui si aggiunge anche un'intestazione molto semplice:

```
\begin{tabular}{|l|rl|}
\hline
Colore          & Numero        &       \\
\hline
nero            & 0     & zero          \\
marrone         & 1     & uno           \\
rosso           & 2     & due           \\
arancio         & 3     & tre           \\
giallo          & 4     & quattro       \\
verde           & 5     & cinque        \\
blu             & 6     & sei           \\
viola           & 7     & sette         \\
```

```
grigio          & 8     & otto          \\
bianco          & 9     & nove          \\
argento         & 10 \% &               \\
oro             & 5 \%  &               \\
                & 2 \%  &               \\
                & 1 \%  &               \\
\hline
\end{tabular}
```

Colore	Numero	
nero	0	zero
marrone	1	uno
rosso	2	due
arancio	3	tre
giallo	4	quattro
verde	5	cinque
blu	6	sei
viola	7	sette
grigio	8	otto
bianco	9	nove
argento	10 %	
oro	5 %	
	2 %	
	1 %	

Per quanto visto fino a questo momento, si intende che non si possono ottenere celle in cui il testo utilizzi più di una riga. Tuttavia, come già mostrato a proposito dell'ambiente '**tabbing**', si può sfruttare il comando '**\parbox**':

```
\documentclass{article}
\usepackage[italian]{babel}
\usepackage[utf8x]{inputenc}
\usepackage[T1]{fontenc}
\frenchspacing
\pagestyle{empty}
\setlength{\textwidth}{14cm}
\begin{document}

\begin{tabular}{|l|rl|l|}
\hline
\parbox{0.2\textwidth}{Colore}
    & \parbox{0.1\textwidth}{Numero}  &   &      \\
\hline
\parbox{0.2\textwidth}{nero}    & \parbox{0.1\textwidth}{0}
    & \parbox{0.2\textwidth}{zero}      &        \\
\parbox{0.2\textwidth}{marrone} & \parbox{0.1\textwidth}{1}
    & \parbox{0.2\textwidth}{uno}       &        \\
\parbox{0.2\textwidth}{rosso}   & \parbox{0.1\textwidth}{2}
    & \parbox{0.2\textwidth}{due}       &        \\
\parbox{0.2\textwidth}{arancio} & \parbox{0.1\textwidth}{3}
    & \parbox{0.2\textwidth}{tre}       &        \\
\parbox{0.2\textwidth}{giallo}  & \parbox{0.1\textwidth}{4}
    & \parbox{0.2\textwidth}{quattro}   &        \\
\parbox{0.2\textwidth}{verde}   & \parbox{0.1\textwidth}{5}
    & \parbox{0.2\textwidth}{cinque}    &        \\
\parbox{0.2\textwidth}{blu}     & \parbox{0.1\textwidth}{6}
    & \parbox{0.2\textwidth}{sei}       &        \\
\parbox{0.2\textwidth}{viola}   & \parbox{0.1\textwidth}{7}
    & \parbox{0.2\textwidth}{sette}     &        \\
\parbox{0.2\textwidth}{grigio}  & \parbox{0.1\textwidth}{8}
    & \parbox{0.2\textwidth}{otto}      &        \\
\parbox{0.2\textwidth}{bianco}  & \parbox{0.1\textwidth}{9}
    & \parbox{0.2\textwidth}{nove}      &        \\

\parbox{0.2\textwidth}{argento}
    & \parbox{0.1\textwidth}{10 \%}
    & & \parbox{0.3\textwidth}{bla bla bla bla bla bla bla
                             bla bla bla bla} \\
\parbox{0.2\textwidth}{oro}
                          & \parbox{0.1\textwidth}{5 \%}
    &   &        \\
                          & \parbox{0.1\textwidth}{2 \%}
    &   &        \\
                          & \parbox{0.1\textwidth}{1 \%}
    &   &        \\
\hline
\end{tabular}
```

```
\end{document}
```

Colore	Numero		
nero	0	zero	
marrone	1	uno	
rosso	2	due	
arancio	3	tre	
giallo	4	quattro	
verde	5	cinque	
blu	6	sei	
viola	7	sette	
grigio	8	otto	
bianco	9	nove	
argento	10 %		bla bla bla bla bla bla bla bla bla bla bla bla bla
oro	5 %		
	2 %		
	1 %		

Per ottenere un risultato molto simile a questo, si potrebbe usare una dichiarazione particolare nella definizione delle colonne:

```
\begin{tabular}{|p{0.2\textwidth}|↩
↪p{0.1}\textwidth}p{0.2\textwidth}|p{0.3\textwidth}|}
```

In tal modo si semplifica l'indicazione dei dati nelle celle:

```
\begin{tabular}{|p{0.2\textwidth}|p{0.1\textwidth}
p{0.2\textwidth}|p{0.3\textwidth}|}
\hline
Colore          & Numero        &       &       \\
\hline
nero            & 0     & zero          &       \\
marrone         & 1     & uno           &       \\
rosso           & 2     & due           &       \\
arancio         & 3     & tre           &       \\
giallo          & 4     & quattro       &       \\
verde           & 5     & cinque        &       \\
blu             & 6     & sei           &       \\
viola           & 7     & sette         &       \\
grigio          & 8     & otto          &       \\
bianco          & 9     & nove          &       \\
argento         & 10 \% &               & bla bla bla bla
                                          bla bla bla bla
                                          bla bla bla \\
oro             & 5 \%  &               &       \\
                & 2 \%  &               &       \\
                & 1 \%  &               &       \\
\hline
\end{tabular}
```

Colore	Numero		
nero	0	zero	
marrone	1	uno	
rosso	2	due	
arancio	3	tre	
giallo	4	quattro	
verde	5	cinque	
blu	6	sei	
viola	7	sette	
grigio	8	otto	
bianco	9	nove	
argento	10 %		bla bla bla bla bla bla bla bla bla bla bla bla bla
oro	5 %		
	2 %		
	1 %		

Come si può intuire, quella '**p{}**' è un modo per dire che il contenuto delle celle deve essere racchiuso all'interno del comando '**\parbox**' e il funzionamento è molto simile.

Oltre alla possibilità di usare il comando '**\hline**' per ottenere una linea orizzontale che divida le righe della tabella, si può ottenere una linea che intervenga solo nell'ambito di alcune colonne, attraverso il comando '**\cline**'. Si osservi l'esempio seguente che è solo una piccola variante dell'ultimo già presentato:

```
\begin{tabular}{|p{0.2\textwidth}|p{0.1\textwidth}
p{0.2\textwidth}|p{0.3\textwidth}|}
\hline
Colore          & Numero &      &       \\
\hline
```

```
nero            & 0      & zero  &       \\
marrone         & 1      & uno   &       \\
…
\cline{1-3}
argento         & 10 \%  &       & & bla bla bla
                                       bla bla bla
                                       bla bla \\
oro             & 5 \%   &       &       \\
\cline{2-4}
                & 2 \%   &       &       \\
                & 1 \%   &       &       \\
\hline
\end{tabular}
```

In pratica, ‘**\cline{1-3}**’ genera una linea orizzontale che parte dall’inizio della prima colonna, fino alla fine della terza colonna; nello stesso modo, ‘**\cline{2-4}**’ genera una linea dalla seconda colonna fino alla quarta inclusa:[4]

Colore	Numero		
nero	0	zero	
marrone	1	uno	
rosso	2	due	
arancio	3	tre	
giallo	4	quattro	
verde	5	cinque	
blu	6	sei	
viola	7	sette	
grigio	8	otto	
bianco	9	nove	
argento	10 %		bla bla bla bla bla bla bla bla bla bla bla bla
oro	5 %		
	2 %		
	1 %		

50.11.3 Tabella a cavallo di più pagine

L’ambiente ‘**tabular**’ genera un blocco che non può essere diviso tra le pagine; per ottenere una tabella che può continuare nelle pagine successive, si può usare l’ambiente ‘**longtable**’, che però richiede un pacchetto esterno con lo stesso nome. Il funzionamento generale dell’ambiente ‘**longtable**’ è molto simile a quello di ‘**tabular**’, con l’aggiunta di qualche comando che consente di individuare un’intestazione e un piede da ripetere quando la tabella si interrompe e riprende nella pagina successiva:

Comando	Descrizione
`\usepackage{longtable}`	dichiara l’utilizzo del pacchetto necessario alla gestione delle tabelle «lunghe»;
`\begin{longtable}{`***descrizione_colonne***`}` … `\end{longtable}`	dichiara l’ambiente di una tabella lunga;
`\endhead`	dichiara la fine delle righe da usare come intestazione;
`\endfoot`	dichiara la fine delle righe da usare alla base della tabella.

In generale, una volta dichiarato l’ambiente, si inseriscono le righe da usare come intestazione e subito dopo quelle da usare alla base della tabella; infine si inseriscono le righe del corpo della tabella.

Per il funzionamento dell’ambiente ‘**longtable**’ non è indispensabile individuare delle righe di intestazione e di conclusione della tabella; tuttavia, per ottenere un risultato gradevole al cambio pagina, conviene sfruttare questa caratteristica.

L’esempio seguente è solo una piccola variante di un altro già apparso a proposito dell’ambiente ‘**tabular**’. Si osservi in particolare l’uso dei comandi ‘**\endhead**’ e ‘**\endfoot**’:


```
…
\usepackage{longtable}
…
\begin{document}
…
\begin{longtable}{|p{0.2\textwidth}|p{0.1\textwidth}
p{0.2\textwidth}|p{0.3\textwidth}|}
\hline
Colore          & Numero &       &       \\
\hline
\endhead
\hline
\endfoot
nero            & 0      & zero  &       \\
marrone         & 1      & uno   &       \\
…
                & 1 \%   &       &       \\
\end{longtable}
```

Come si vede nell’esempio, ciò che viene messo alla base della tabella è semplicemente un comando ‘**\hline**’, per ottenere una linea orizzontale quando la tabella si interrompe; così facendo, alla fine della tabella non viene aggiunto un comando ‘**\hline**’, perché altrimenti i due si sommerebbero:

Colore	Numero		
nero	0	zero	
marrone	1	uno	
rosso	2	due	
arancio	3	tre	
giallo	4	quattro	
verde	5	cinque	
blu	6	sei	
viola	7	sette	
grigio	8	otto	
bianco	9	nove	
argento	10 %		bla bla bla bla bla bla bla bla bla bla bla bla
oro	5 %		
	2 %		
	1 %		

Le tabelle ottenute con l’ambiente ‘**longtable**’ hanno degli inconvenienti. Quello più appariscente riguarda l’incapacità di tenere in considerazione le note a piè di pagina, le quali vengono tranquillamente sovrascritte da una tabella che scorre anche sulla pagina successiva.

50.12 LaTeX: figure

Esistono diversi modi di realizzare dei disegni o di importare delle immagini esterne con LaTeX. Probabilmente, la tecnica più comune è quella di incorporare un’immagine o un disegno realizzato con strumenti interattivi esterni, ma è comunque importante conoscere più possibilità a disposizione, anche se non nel dettaglio.

50.12.1 Disegnare con LaTeX

LaTeX offre un ambiente particolare, ‘**picture**’, nel quale è possibile realizzare dei disegni utilizzando eventualmente dei comandi specifici. In generale, si tratta soprattutto di grafici, che contengono anche del testo collocato in modo preciso. L’esempio seguente serve per dare un punto di riferimento visivo alla descrizione che viene fatta successivamente a proposito di questo ambiente:

```
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 bla
bla bla bla

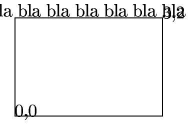
\setlength{\unitlength}{1cm}
\begin{picture}(3,2)(0,0)
\put(0,0){0,0}
\put(0,0){\line(1,0){3}}
\put(0,2){\line(1,0){3}}
\put(0,0){\line(0,1){2}}
\put(3,0){\line(0,1){2}}
```

```
\put(3,2){3,2}
\end{picture}

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 bla
bla bla bla
```

Ecco il risultato della composizione. Il testo prima e dopo la figura, serve a comprendere in che modo LaTeX la allinea e quanto viene spaziata verticalmente:

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 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 bla bla bla bla bla bla bla bla bla

L'ambiente '**picture**' viene dichiarato in modo insolito rispetto agli altri comandi di LaTeX:

```
\setlength{\unitlength}{dimensione}

...

\begin{picture}(ampiezza,altezza)(coordinata_iniziale_x,↩
↪          coordinata_iniziale_y)
    \put(coordinata_x,coordinata_y){oggetto}
    ...
    ...
\end{picture}
```

In pratica, da qualche parte del documento, prima dell'uso dell'ambiente '**picture**', occorre assegnare un valore (non elastico) alla lunghezza '**\unitlength**'; questa lunghezza diventa poi l'unità di misura delle lunghezze e delle coordinate usate nell'ambito dell'ambiente '**picture**'. Naturalmente, il valore di '**\unitlength**' può essere cambiato successivamente, prima di altre figure.

Come si può vedere, l'ambiente '**picture**' utilizza degli argomenti tra parentesi tonde, che non si trovano altrimenti in LaTeX, ma la cosa più interessante è che la seconda parentesi tonda è facoltativa.[5]

La prima coppia di numeri tra parentesi tonde serve a delimitare un rettangolo ideale, all'interno del quale si intende collocare la figura. Nel caso dell'esempio, dal momento che '**\unitlength**' è di 1 cm, si tratta di un rettangolo largo 3 cm e alto 2 cm. La seconda coppia di numeri, opzionale, specifica le coordinate dell'angolo inferiore sinistro. Di solito si assegna a queste coordinate il valore zero, come si vede nell'esempio.

> Gli «oggetti» grafici o testuali che vengono piazzati nella figura possono anche essere collocati al di fuori del rettangolo che la dovrebbe contenere, ma ovviamente, in tal modo viene meno il senso della definizione all'inizio dell'ambiente '**picture**'.

Tutto ciò che appare all'interno dell'ambiente '**picture**' viene inserito tramite il comando '**\put**', dove, sempre tra parentesi quadre, si individuano le coordinate iniziali di un oggetto da collocare. Queste coordinata iniziali si intendono sempre relative alle coordinate iniziali del rettangolo in cui si colloca la figura.

L'esempio introduttivo, mostra un riquadro, disegnato sul bordo esterno del rettangolo che rappresenta l'area della figura. Inoltre, vengono collocate due etichette testuali, a partire dall'angolo inferiore sinistro di questo rettangolo e a partire dall'angolo superiore destro, per consentire di comprendere in che modo viene collocato il testo. In effetti, si può vedere che quanto scritto a partire dall'angolo superiore destro si trova in pratica al di fuori del rettangolo della figura.

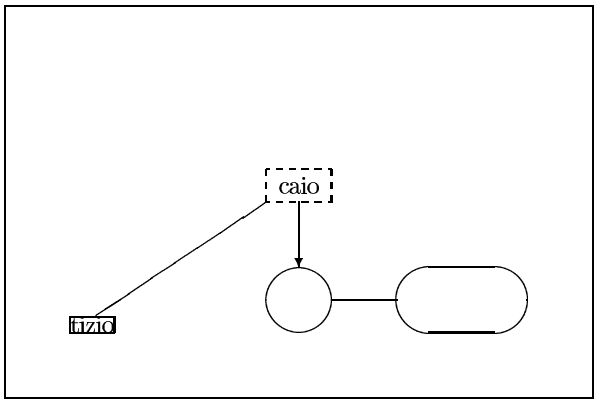
All'interno del controllo di '**\put**' possono essere usati diversi comandi specifici, con i quali si realizzano facilmente linee, riquadri, ovali e cerchi. Viene mostrato un elenco parziale di questi.

<table>
<tr><th>Comando</th><th>Descrizione</th></tr>
<tr><td>\circle*{diametro}</td><td>Disegna un cerchio con centro nelle coordinate iniziali, del diametro indicato, espresso in quantità di '\unitlength'. Se si usa l'asterisco, si ottiene un cerchio pieno (nero).</td></tr>
<tr><td>\dashbox{lunghezza_del_tratteggio}↩
↪(ampiezza,altezza)↩
↪[t|l|b|r|tl|bl|br|tr]{testo}</td><td>Disegna un rettangolo tratteggiato, con le coordinate iniziali corrispondenti all'angolo inferiore sinistro. L'ampiezza e l'altezza sono numeri che esprimono quantità di '\unitlength'. All'interno del rettangolo si può collocare del testo, contenuto nell'ultimo argomento, che di solito appare al centro, salvo il suo riposizionamento con l'uso di lettere particolari nell'argomento opzionale tra parentesi quadre.</td></tr>
<tr><td>\makebox(ampiezza,altezza)↩
↪[t|l|b|r|tl|bl|br|tr]{testo}</td><td>Delimita un rettangolo, senza bordo, con le coordinate iniziali corrispondenti all'angolo inferiore sinistro. L'ampiezza e l'altezza sono numeri che esprimono quantità di '\unitlength'. All'interno del rettangolo si può collocare del testo, contenuto nell'ultimo argomento, che di solito appare al centro, salvo il suo riposizionamento con l'uso di lettere particolari nell'argomento opzionale tra parentesi quadre.</td></tr>
<tr><td>\frame{oggetto}</td><td>Disegna una cornice attorno all'oggetto, senza aggiungere altro spazio.</td></tr>
<tr><td>\line(scostamento_x,scostamento_y)↩
↪{lunghezza}</td><td>Disegna una linea a partire dalle coordinate iniziali, passante idealmente nelle coordinate descritte tra parentesi tonde, come valore che va a sommarsi alle coordinate iniziali. Questo punto passante può essere indicato solo con valori interi, da zero a sei, e serve in pratica a descrivere l'inclinazione della linea.</td></tr>
<tr><td>\vector(scostamento_x,scostamento_y)↩
↪{lunghezza}</td><td>Disegna una linea, come il comando '\line', aggiungendo una freccia all'estremità finale.</td></tr>
<tr><td>\oval(ampiezza,altezza)↩
↪[t|l|b|r|tl|bl|br|tr]</td><td>Disegna un ovale, oppure, se molto grande, un rettangolo smussato. Le lettere tra parentesi quadre, opzionali, consentono di mostrare solo una porzione dell'ovale.</td></tr>
</table>

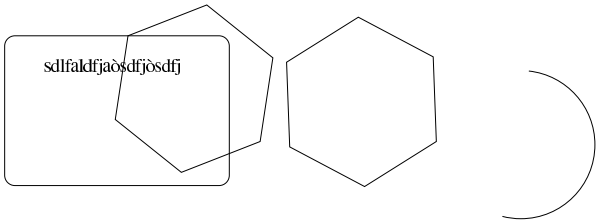
Comando	Descrizione
`\linethickness{`*lunghezza*`}`	Consente di fissare lo spessore delle linee dritte generate da alcuni comandi. È importante osservare che la lunghezza della linea deve essere un valore positivo, espresso in modo indipendente da '**`\unitlength`**'.

Viene mostrato un esempio complessivo dei comandi descritti (escluso '**`\linethickness`**'):

```
\setlength{\unitlength}{1cm}
\begin{picture}(9,6)(0,0)
\put(0,0){\frame{\makebox(9,6){}}}
\put(1,1){\frame{tizio}}
\put(4,3){\dashbox{0.1}(1,0.5){caio}}
\put(4,3){\line(-3,-2){2.6}}
\put(4.5,3){\vector(0,-1){1}}
\put(4.5,1.5){\circle{1}}
\put(7,1.5){\oval(2,1)}
\put(5,1.5){\line(1,0){1}}
\end{picture}
```



Come si intende, la realizzazione di disegni attraverso l'ambiente '**`picture`**', può risultare impegnativo, quando il disegno diventa complesso. Tuttavia esiste la possibilità di usare XFig (sezione 29.6.4), che è in grado di esportare i propri disegni in formato LaTeX, con le limitazioni grafiche necessarie. A titolo di esempio, si suppone di avere realizzato il file FIG '`disegno.fig`', corrispondente a ciò che si può vedere sotto:



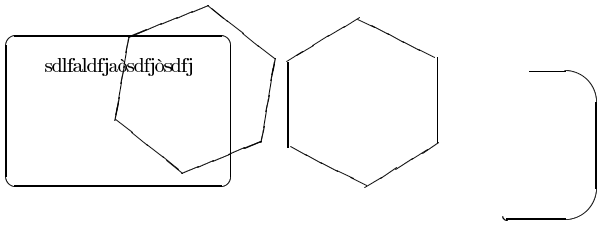
Per trasformare questo file in codice adatto a un sorgente LaTeX si può usare '**`fig2dev`**' nel modo seguente, allo scopo di generare il file '`disegno.tex`':

```
$ fig2dev -L latex disegno.fig disegno.tex [Invio]
```

Il file che si ottiene è piuttosto complesso e probabilmente qualche comando non viene individuato correttamente da LaTeX. Per ovviare all'inconveniente, dovrebbe essere sufficiente eliminare i comandi che creano dei problemi. In generale dovrebbe trattarsi del comando '**`\color`**':

```
$ cat disegno.tex ↩
↪| sed "s/[\]color\\[rgb\\][{][0-9,]*[}]//" ↩
↪> disegno_2.tex [Invio]
```

Ecco cosa si ottiene alla fine, con la composizione LaTeX:



50.12.2 Importazione di file EPS esterni

Un modo più semplice di inserire delle immagini nel testo realizzato con LaTeX è quello di incorporare delle immagini esterne attraverso il pacchetto '**`epsfig`**', il quale si incorpora con il comando seguente, inserito preferibilmente nel preambolo:

```
\usepackage{epsfig}
```

Successivamente è possibile usare il comando '**`\epsfig`**' per fare riferimento a un file EPS (*Encapsulated PostScript*) da inserire nella composizione finale. Semplificando si può indicare come nel modello sintattico seguente:

```
\epsfig{file=file_eps,height=altezza,width=larghezza,↩
↪angle=rotazione}
```

Il file rappresenta il percorso assoluto o relativo di un file EPS (oppure PostScript), ma senza estensione: il pacchetto '**`epsfig`**' si attende di trovarlo con l'estensione '`.ps`'. La larghezza e l'altezza vengono espresse nell'unità di misura desiderata; infine, l'angolo di rotazione permette di girare l'immagine, ma di solito si lascia il valore zero.

Non tutte le informazioni sono necessarie; per esempio, se si omettono le informazioni su altezza e larghezza, si fa riferimento alle dimensioni originali; nello stesso modo, se si mette solo la larghezza, si ottiene un'altezza proporzionale. In pratica, si individuano delle opzioni separate da virgole che possono essere omesse.

L'uso del pacchetto '**`epsfig`**' è compatibile con pdfLaTeX, il quale invece incorpora immagini in formato PNG. Questi file vengono cercati nello stesso percorso, ma con estensione '`.png`'.

In alternativa esiste anche un altro pacchetto adatto all'inserimento di immagini EPS. Si tratta di '**`graphicx`**' che si incorpora con il comando seguente inserito nel preambolo del sorgente LaTeX:

```
\usepackage{graphicx}
```

Successivamente si può usare il comando '**`\includegraphics`**':

```
\includegraphics↩
↪[width=larghezza,height=altezza,scale=scala,angle=rotazione]↩
↪{file_eps}
```

A differenza di quanto visto con '**`\epsfig`**', c'è anche la possibilità di definire una scala, attraverso un valore, dove l'unità rappresenta un rapporto pari al 100 %.

Con '**`\includegraphics`**' esiste anche una differenza più importante rispetto a '**`\epsfig`**': è necessario indicare anche l'estensione del file, la quale non è più determinata automaticamente. In questo modo, se si intende usare pdfLaTeX, è necessario fare riferimento a formati grafici differenti, cambiando di conseguenza l'estensione.

50.12.3 Estrazione di file EPS dal file PostScript finale

All'interno di un file PostScript, generato attraverso LaTeX e Dvips, le immagini inserite incorporando file EPS, come descritto nella sezione precedente, sono delimitate da due direttive molto precise:

```
%%BeginDocument: nome_file
codice_eps_della_figura
%%EndDocument
```

Sapendo questo, è possibile recuperare le immagini da un file PostScript di questo tipo attraverso un procedimento manuale, o automatico. Quello che segue è un piccolo programma Perl che legge dallo standard input il file PostScript e crea tutti i file che trova dichiarati come appena descritto:

```
#!/usr/bin/perl
#
my ($line)    = "";
my ($picture) = "";
my ($save)    = 0;
#
while ($line = <STDIN>)
  {
    if    ($line =~ m/^\%\%BeginDocument: (\S+)$/)
      {
        $save = 1;
        $picture = $1;
        open (PICTURE, "> $picture");
      }
    elsif ($line =~ m/^\%\%EndDocument$/)
      {
        close (PICTURE);
        $picture = "";
        $save = 0;
      }
    else
      {
        if ($save)
          {
            print PICTURE ("$line");
          }
      }
  }
```

Supponendo che questo programma sia stato salvato nel file '`ps-latex-picture-save`' e che sia stato reso eseguibile, si potrebbe usare come nell'esempio seguente:

```
$ ps-latex-picture-save < mio_file.ps [Invio]
```

Si osservi che questo procedimento può funzionare solo se il file PostScript di partenza è stato ottenuto direttamente da Dvips, tenendo conto che non ci devono essere state delle rielaborazioni successive, altrimenti si perdono i riferimenti descritti.

50.13 Ambienti matematici

Come avviene per TeX, anche LaTeX distingue una modalità di composizione normale, da una modalità matematica. A differenza di TeX, esistono degli ambienti speciali per la dichiarazione della modalità matematica, ma è ancora ammissibile l'uso del simbolo '**$**' e di '**$$**' per individuare l'inizio e la fine degli ambienti matematici.

Molto di quanto già descritto a proposito di TeX (sezione 49.16), vale ovviamente anche per la composizione matematica di LaTeX. Pertanto, alcuni dettagli non vengono descritti nuovamente.

Si tenga presente che la gestione degli ambienti matematici potrebbe essere ancora più raffinata con l'utilizzo del pacchetto '**amsmath**' ('**\usepackage{amsmath}**').

50.13.1 Definizione della modalità matematica

La modalità matematica si può definire con LaTeX in diversi modi. Anche con LaTeX si distingue se la composizione matematica vada inserita nell'ambito del testo normale, oppure in un blocco indipendente. L'elenco seguente riepiloga le istruzioni necessarie a delimitare gli ambienti matematici:

Comando	Descrizione

`\begin{math}`*testo_matematico*`\end{math}`
`\(`*testo_matematico*`\)`
`$`*testo_matematico*`$`
	compone il testo matematico nel blocco già esistente;

`\begin{displaymath}`
testo_matematico
`\end{displaymath}`
`\[`*testo_matematico*`\]`
`$$`*testo_matematico*`$$`
	compone il testo matematico in un blocco separato;

`\begin{equation}`
testo_matematico
`\end{equation}`
	compone il testo matematico in un blocco separato, annotando sul bordo il numero dell'equazione.

Come si vede dall'elenco, vengono offerte delle alternative per ottenere lo stesso tipo di ambiente. Vengono mostrati due esempi molto semplici, già apparsi nel capitolo riferito a TeX, che riguardano l'inserimento di una formula nel testo normale e la sua separazione in un blocco a sé stante:

```
Se $R$ è il ritardo di ogni singola consegna, $T$
è il tempo a disposizione per la consegna, $I$ è il valore
della merce, il ritardo medio si esprime come
\begin{math}{\sum {R \over T} \cdot I}\over{\sum I}
\end{math}.
```

Se R è il ritardo di ogni singola consegna, T è il tempo a disposizione per la consegna, I è il valore della merce, il ritardo medio si esprime come $\frac{\sum \frac{R}{T} \cdot I}{\sum I}$.

```
Se $R$ è il ritardo di ogni singola consegna, $T$
è il tempo a disposizione per la consegna, $I$ è il valore
della merce, il ritardo medio si esprime secondo la formula
seguente:

\begin{displaymath}{\sum {R \over T} \cdot I}\over{\sum
I}\end{displaymath}
```

Se R è il ritardo di ogni singola consegna, T è il tempo a disposizione per la consegna, I è il valore della merce, il ritardo medio si esprime secondo la formula seguente:

$$\frac{\sum \frac{R}{T} \cdot I}{\sum I}$$

A parte le situazioni normali, merita attenzione l'ambiente '**equation**', con il quale si può inserire un'equazione in un blocco separato, al quale si può fare riferimento tramite un'etichetta. Si osservi l'esempio:

```
L'equazione \ref{ritardo-medio} mostra in che modo si
calcola il ritardo medio, dove $R$ è il ritardo di ogni
singola consegna, $T$ è il tempo a disposizione per la
consegna e $I$ è il valore della merce.

\begin{equation}\label{ritardo-medio}
{\sum {R \over T} \cdot I}\over{\sum I}
\end{equation}
```

L'equazione 1 mostra in che modo si calcola il ritardo medio, dove R è il ritardo di ogni singola consegna, T è il tempo a disposizione per la consegna e I è il valore della merce.

$$\frac{\sum \frac{R}{T} \cdot I}{\sum I} \tag{1}$$

50.13.2 Spazi orizzontali e punteggiatura

LaTeX si comporta nello stesso modo di TeX a proposito della spaziatura orizzontale negli ambienti matematici. In pratica, gli spazi orizzontali vengono ignorati, a meno che si usino comandi appositi per inserirli:

Sequenza	Significato
`\qquad`	Due quadratoni.
`\quad`	Un quadratone.

Sequenza	Significato
`\<SP>`	Uno spazio «normale».
`$\;$`	Uno spazio spesso (5/8 di quadratone).
`$\>$`	Uno spazio medio (2/9 di quadratone).
`$\.$`	Uno spazio molto sottile (1/6 di quadratone).
`$\!$`	Uno spazio negativo (-1/6 di quadratone).

Per inserire del testo descrittivo all'interno di un ambiente matematico, lo si può delimitare in una scatola orizzontale, con il comando '**`\makebox`**', o più semplicemente con '**`\mbox`**':

```
$prima durante e dopo$\par
$prima \mbox{ durante } e dopo$\par
```

Come si può intendere, si vuole isolare e spaziare la parola «durante» rispetto al resto, mostrandola con caratteri normali.

$$primaduranteedopo$$
$$prima\ \text{durante}\ edopo$$

Anche la spaziatura attorno ai simboli di punteggiatura funziona esattamente come con TeX puro e semplice. L'esempio seguente riassume i casi principali:

```
$123456.7890$\par
$123456,7890$\par
$123456{,}7890$\par
$123456;7890$\par
$123456:7890$\par
$123456:=7890$\par
$123456\colon 7890$\par
```

$$123456.7890$$
$$123456,7890$$
$$123456{,}7890$$
$$123456;7890$$
$$123456:7890$$
$$123456:=7890$$
$$123456\colon 7890$$

50.13.3 Caratteri e simboli

LaTeX è più ricco di simboli rispetto a TeX, se si utilizza il pacchetto '**`latexsym`**':

```
\usepackage{latexsym}
```

Le tabelle che seguono riepilogano i simboli a disposizione, inclusi quelli provenienti dal pacchetto '**`latexsym`**'.

Tabella 50.282. Accenti nell'ambiente matematico.

Simbolo	Codice	Annotazioni
$\acute{o}$	$\acute o$	Accento acuto matematico.
$\grave{o}$	$\grave o$	Accento grave matematico.
$\hat{o}$	$\hat o$	Accento circonflesso matematico.
$\ddot{o}$	$\ddot o$	
$\tilde{o}$	$\tilde o$	
$\bar{o}$	$\bar o$	
$\dot{o}$	$\dot o$	
$\breve{o}$	$\breve o$	
$\check{o}$	$\check o$	
$\vec{o}$	$\vec o$	
$\imath$	$\imath$	Da usare per aggiungere un accento.
$\jmath$	$\jmath$	Da usare per aggiungere un accento.
$\widehat{abc}$	$\widehat{abc}$	
$\widetilde{abc}$	$\widetilde{abc}$	

Tabella 50.283. Lettere greche.

Simbolo	Codice	Simbolo	Codice	Simbolo	Codice	Nome
α	α			A	\textrm{A}	Alfa
β	β			B	\textrm{B}	Beta
γ	γ			Γ	Γ	Gamma
δ	δ			Δ	Δ	Delta
ϵ	ϵ	ε	ε	E	\textrm{E}	Epsilon
ζ	ζ			Z	\textrm{Z}	Zeta
η	η			H	\textrm{H}	Eta
θ	θ	ϑ	ϑ	Θ	Θ	Theta
ι	ι			I	\textrm{I}	Iota
κ	κ			K	\textrm{K}	Kappa
λ	λ			Λ	Λ	Lambda
μ	μ			M	\textrm{M}	Mu
ν	ν			N	\textrm{N}	Nu
ξ	ξ			Ξ	Ξ	Xi
o	\textrm{o}	o	o	O	\textrm{O}	Omicron
π	π	ϖ	ϖ	Π	Π	Pi
ρ	ρ	ϱ	ϱ	P	\textrm{P}	Rho
σ	σ	ς	ς	Σ	Σ	Sigma
τ	τ			T	\textrm{T}	Tau
υ	υ			Υ	Υ	Upsilon
ϕ	ϕ	φ	φ	Φ	Φ	Phi
χ	χ			X	\textrm{X}	Chi
ψ	ψ			Ψ	Ψ	Psi
ω	ω			Ω	Ω	Omega

Tabella 50.284. Simboli matematici comuni.

Simbolo	Codice	Annotazioni
$\backslash$	$\backslash$	Barra obliqua inversa.
$\aleph$	$\aleph$	
$\hbar$	$\hbar$	
ℓ	ℓ	
$\wp$	$\wp$	
$\Re$	$\Re$	
$\Im$	$\Im$	
∂	∂	Derivata parziale.
∞	∞	Infinito.
$\prime$	$\prime$	Primo.
$'$	$'$	
$\emptyset$	$\emptyset$	
∇	∇	
$\top$	$\top$	
$\bot$	$\bot$	
$\vert$	$\vert$	
$\mid$	$\mid$	Equivalente a $\vert$.
$\Vert$	$\Vert$	
$\parallel$	$\parallel$	Equivalente a $\Vert$.
$\angle$	$\angle$	
$\triangle$	$\triangle$	
$\forall$	$\forall$	
$\exists$	$\exists$	
$\neg$	$\neg$	Equivalente a $\lnot$.
$\flat$	$\flat$	
$\natural$	$\natural$	
$\sharp$	$\sharp$	
$\surd$	$\surd$	Radice senza la linea superiore.
$\clubsuit$	$\clubsuit$	Fiori.
$\diamondsuit$	$\diamondsuit$	Quadri.
$\heartsuit$	$\heartsuit$	Cuori.
$\spadesuit$	$\spadesuit$	Picche.
$\mho$	$\mho$	Richiede il pacchetto `latexsym`.
$\Box$	$\Box$	"
$\Diamond$	$\Diamond$	"

Tabella 50.285. Operatori binari comuni.

Simbolo	Codice	Annotazioni
$\pm$	$\pm$	Più o meno.
$\mp$	$\mp$	Meno o più.
$\cdot$	$\cdot$	
$\setminus$	$\setminus$	
$+$	$+$	Somma.
$-$	$-$	Sottrazione.
$\times$	$\times$	Moltiplicazione.
$\div$	$\div$	Divisione.
	$*$	Moltiplicazione discrezionale.
$*$	$\ast$	Asterisco.
$\star$	$\star$	Stella.
$\diamond$	$\diamond$	
$\circ$	$\circ$	
$\bullet$	$\bullet$	
$\cap$	$\cap$	Intersezione.
$\cup$	$\cup$	Unione.
$\uplus$	$\uplus$	
$\sqcap$	$\sqcap$	
$\sqcup$	$\sqcup$	
$\triangleleft$	$\triangleleft$	
$\triangleright$	$\triangleright$	
$\wr$	$\wr$	
$\bigcirc$	$\bigcirc$	
$\bigtriangleup$	$\bigtriangleup$	
$\bigtriangledown$	$\bigtriangledown$	
$\vee$	$\vee$	
$\vee$	$\lor$	OR logico (uguale a $\vee$).
$\wedge$	$\wedge$	
$\wedge$	$\land$	AND logico (uguale a $\wedge$).
$\oplus$	$\oplus$	
$\ominus$	$\ominus$	
$\otimes$	$\otimes$	
$\oslash$	$\oslash$	
$\odot$	$\odot$	
$\dagger$	$\dagger$	
$\ddagger$	$\ddagger$	
$\amalg$	$\amalg$	
$\lhd$	$\lhd$	Richiede il pacchetto `latexsym`.
$\rhd$	$\rhd$	”
$\unlhd$	$\unlhd$	”
$\unrhd$	$\unrhd$	”

Tabella 50.286. Operatori di relazione.

Simbolo	Codice	Simbolo	Codice
$=$	$=$	$\neq$	$\not =$
		$\neq$	$\neq$
		$\neq$	$\ne$
$\equiv$	$\equiv$	$\not\equiv$	$\not \equiv$
$\sim$	$\sim$	$\not\sim$	$\not \sim$
$\simeq$	$\simeq$	$\not\simeq$	$\not \simeq$
$\approx$	$\approx$	$\not\approx$	$\not \approx$
$\cong$	$\cong$	$\not\cong$	$\not \cong$
$\asymp$	$\asymp$	$\not\asymp$	$\not \asymp$
$\bowtie$	$\bowtie$		
$\doteq$	$\doteq$		
$\parallel$	$\parallel$		
$\perp$	$\perp$	$\not\perp$	$\not \perp$
$<$	$<$	$\not<$	$\not <$
$\leq$	$\leq$	$\not\leq$	$\not \leq$
$\leq$	$\le$	$\not\leq$	$\not \le$
$>$	$>$	$\not>$	$\not >$
$\geq$	$\geq$	$\not\geq$	$\not \geq$
$\geq$	$\ge$	$\not\geq$	$\not \ge$
$\prec$	$\prec$	$\not\prec$	$\not \prec$
$\preceq$	$\preceq$	$\not\preceq$	$\not \preceq$
$\succ$	$\succ$	$\not\succ$	$\not \succ$
$\succeq$	$\succeq$	$\not\succeq$	$\not \succeq$
$\ll$	$\ll$		
$\gg$	$\gg$		
$\subset$	$\subset$	$\not\subset$	$\not \subset$
$\subseteq$	$\subseteq$	$\not\subseteq$	$\not \subseteq$
$\supset$	$\supset$	$\not\supset$	$\not \supset$
$\supseteq$	$\supseteq$	$\not\supseteq$	$\not \supseteq$
$\sqsubseteq$	$\sqsubseteq$	$\not\sqsubseteq$	$\not \sqsubseteq$
$\sqsupseteq$	$\sqsupseteq$	$\not\sqsupseteq$	$\not \sqsupseteq$
$\mid$	$\mid$		
$\smile$	$\smile$		
$\frown$	$\frown$		
$\in$	$\in$	$\notin$	$\not \in$
$\ni$	$\ni$	$\not\ni$	$\not \ni$
$\owns$	$\owns$	$\not\owns$	$\not \owns$
$\vdash$	$\vdash$		
$\dashv$	$\dashv$		
$\models$	$\models$		
$\propto$	$\propto$		
$\sqsubset$	$\sqsubset$	$\not\sqsubset$	$\not \sqsubset$
$\sqsupset$	$\sqsupset$	$\not\sqsupset$	$\not \sqsupset$
$\bowtie$	$\Join$		

Tabella 50.287. Frecce.

Simbolo	Codice	Simbolo	Codice
$\leftarrow$	$\leftarrow$	$\rightarrow$	$\rightarrow$
$\leftarrow$	$\gets$	$\rightarrow$	$\to$
$\Leftarrow$	$\Leftarrow$	$\Rightarrow$	$\Rightarrow$
$\longleftarrow$	$\longleftarrow$	$\longrightarrow$	$\longrightarrow$
$\Longleftarrow$	$\Longleftarrow$	$\Longrightarrow$	$\Longrightarrow$
$\hookleftarrow$	$\hookleftarrow$	$\hookrightarrow$	$\hookrightarrow$
$\leftharpoonup$	$\leftharpoonup$	$\rightharpoonup$	$\rightharpoonup$
$\leftharpoondown$	$\leftharpoondown$	$\rightharpoondown$	$\rightharpoondown$
$\rightleftharpoons$	$\rightleftharpoons$		
$\iff$	$\iff$		
$\mapsto$	$\mapsto$		
$\longmapsto$	$\longmapsto$		
$\leftrightarrow$	$\leftrightarrow$	$\Leftrightarrow$	$\Leftrightarrow$
$\longleftrightarrow$	$\longleftrightarrow$	$\Longleftrightarrow$	$\Longleftrightarrow$
$\uparrow$	$\uparrow$	$\downarrow$	$\downarrow$
$\Uparrow$	$\Uparrow$	$\Downarrow$	$\Downarrow$
$\updownarrow$	$\updownarrow$	$\Updownarrow$	$\Updownarrow$
$\nwarrow$	$\nwarrow$	$\nearrow$	$\nearrow$
$\swarrow$	$\swarrow$	$\searrow$	$\searrow$
$\overleftarrow{ab}$	$\overleftarrow {ab}$	$\overrightarrow{ab}$	$\overrightarrow {ab}$
$\vec{ab}$	$\vec {ab}$		
$\leadsto$	$\leadsto$		

Tabella 50.288. Ellissi.

Simbolo	Codice	Annotazioni
$\cdots$	$\cdots$	Ellissi orizzontale al centro.
$\ldots$	$\ldots$	Ellissi orizzontale in basso.
$\vdots$	$\vdots$	Ellissi verticale.
$\ddots$	$\ddots$	Ellissi diagonale.

Tabella 50.289. Altri simboli matematici.

Simbolo	Codice	Annotazioni
$\sum$	$\sum$	Sommatoria.
$\prod$	$\prod$	
$\coprod$	$\coprod$	
$\int$	$\int$	Integrale.
$\oint$	$\oint$	
$\bigcap$	$\bigcap$	Intersezione.
$\bigcup$	$\bigcup$	Unione.
$\bigsqcup$	$\bigsqcup$	
$\bigvee$	$\bigvee$	
$\bigwedge$	$\bigwedge$	
$\bigoplus$	$\bigoplus$	
$\biguplus$	$\biguplus$	
$\bigotimes$	$\bigotimes$	
$\bigodot$	$\bigodot$	

50.13.4 Delimitatori

LaTeX dispone degli stessi delimitatori di TeX, mantenendo la disponibilità dei comandi ‘`\big`...’ e ‘`\Big`...’ per gli ingrandimenti controllati. La tabella seguente serve a richiamare l’uso di questi delimitatori, tenendo conto che l’uso dei comandi ‘`\big`...’ e ‘`\Big`...’ può essere esteso anche oltre gli esempi mostrati.

Tabella 50.290. Delimitatori.

	Codice			Codice	
( )	$($ $)$				
( )	$\bigl($	$\bigr)$	$\Bigl($	$\Bigr)$	
$\left(\right)$	$\biggl($	$\biggr)$	$\Biggl($	$\Biggr)$	
[ ]	$[$	$]$	$\lbrack$	$\rbrack$	
[ ]	$\bigl[$	$\bigr]$	$\Bigl[$	$\Bigr]$	
$\left[\right]$	$\biggl[$	$\biggr]$	$\Biggl[$	$\Biggr]$	
{ }	$\{$	$\}$	$\lbrace$	$\rbrace$	
{ }	$\bigl\{$	$\bigr\}$	$\Bigl\{$	$\Bigr\}$	
$\left\{ \right\}$	$\biggl\{$	$\biggr\}$	$\Biggl\{$	$\Biggr\}$	
⟨ ⟩	$\langle$	$\rangle$			
⟨ ⟩	$\bigl\langle$	$\bigr\rangle$	$\Bigl\langle$	$\Bigr\rangle$	
$\left\langle \right\rangle$	$\biggl\langle$	$\biggr\rangle$	$\Biggl\langle$	$\Biggr\rangle$	
⌊ ⌋	$\lfloor$	$\rfloor$			
⌊ ⌋	$\bigl\lfloor$	$\bigr\rfloor$	$\Bigl\lfloor$	$\Bigr\rfloor$	
$\left\lfloor \right\rfloor$	$\biggl\lfloor$	$\biggr\rfloor$	$\Biggl\lfloor$	$\Biggr\rfloor$	
⌈ ⌉	$\lceil$	$\rceil$			
⌈ ⌉	$\bigl\lceil$	$\bigr\rceil$	$\Bigl\lceil$	$\Bigr\rceil$	
$\left\lceil \right\rceil$	$\biggl\lceil$	$\biggr\rceil$	$\Biggl\lceil$	$\Biggr\rceil$	
(x)	$\left(x\right)$		$\left(\frac{x}{y}\right)$	$\left(x\over y\right)$	
/	$/$		\	$\backslash$	
∣ ‖	$\|$	$\Vert$	‖	$\|$	
↑ ↓	$\uparrow$	$\downarrow$	⇑ ⇓	$\Uparrow$	$\Downarrow$
↕	$\updownarrow$		⇕	$\Updownarrow$	

Esiste comunque la possibilità di usare comandi '**\big**...' '**\Big**...', senza la lettera '**l**' o '**r**' che specifica trattarsi di un delimitatore sinistro o destro:

	Codice			Codice	
( )	$\big($	$\big)$	$\Big($	$\Big)$	
$\left(\right)$	$\bigg($	$\bigg)$	$\Bigg($	$\Bigg)$	

LaTeX offre anche qualche delimitatore speciale; precisamente delle parentesi graffe orizzontali, da usare sopra o sotto qualcosa:

Comando	Descrizione
`\overbrace{`*testo*`}`	sovrappone una parentesi graffa sopra il testo;
`\underbrace{`*testo*`}`	sovrappone una parentesi graffa sopra il testo.

Sono ancora validi con LaTeX i comandi '**\overline**' e '**\underline**', per sopralineare e sottolineare nell'ambiente matematico.

50.13.5 Modifica delle caratteristiche del carattere

Anche con LaTeX sono disponibili i comandi che permettono di controllare la dimensione del carattere negli ambienti matematici. In ordine decrescente, si tratta di: '**\displaystyle**', '**\textstyle**', '**\scriptstyle**' e '**\scriptscriptstyle**'. Viene mostrato un esempio già apparso a proposito di TeX:

```
$$x+{y\over x+{y\over x+{y\over x+{y\over x+{y\over
z}}}}}$$\par
$$x+{y\over \displaystyle x+{y\over \displaystyle x+{y\over
\displaystyle
x+{y\over \displaystyle x+{y\over z}}}}}$$\par
```

$$x+\frac{y}{x+\frac{y}{x+\frac{y}{x+\frac{y}{x+\frac{y}{z}}}}}$$

$$x+\frac{y}{x+\frac{y}{x+\frac{y}{x+\frac{y}{x+\frac{y}{z}}}}}$$

A differenza di TeX, LaTeX aggiunge dei comandi cumulabili, anche per la definizione dello stile e della forma del carattere matematico. Nella sezione 50.7.3 è già stato trattato questo argomento, comunque viene riportato nuovamente l'elenco di questi comandi:

Comando	Descrizione
`\mathnormal{`*testo*`}`	testo matematico normale;
`\mathrm{`*testo*`}`	tondo;
`\mathbf{`*testo*`}`	neretto;
`\mathsf{`*testo*`}`	lineare;
`\mathtt{`*testo*`}`	dattilografico;
`\mathit{`*testo*`}`	corsivo;
`\mathcal{`*testo*`}`	calligrafico;
`\mathsc{`*testo*`}`	maiuscoletto;

50.13.6 Espressioni varie

LaTeX consente di usare i comandi per la realizzazione di frazioni e simili, già disponibili con TeX:

- `$${x+y \over z}$$`
$$\frac{x+y}{z}$$
- `$${x \over y+z}$$`
$$\frac{x}{y+z}$$
- `$${a \over x} + {b \over x} = {a+b \over x}$$`
$$\frac{a}{x}+\frac{b}{x}=\frac{a+b}{x}$$
- `$${x+y \atop z}$$`
$$\begin{matrix}x+y\\z\end{matrix}$$
- `$${x+y \choose z}$$`
$$\binom{x+y}{z}$$
- `$${x+y \above 1mm z}$$`
$$\begin{matrix}x+y\\\hline z\end{matrix}$$

Inoltre, LaTeX mette a disposizione un comando aggiuntivo per le frazioni, che forse può risultare comodo in alcune situazioni:

Comando	Descrizione
`\frac{`*numeratore*`}{`*denominatore*`}`	frazione normale.

Gli apici e i pedici funzionano con LaTeX esattamente come in TeX:

- `$$10^2$$`
$$10^2$$
- `$$10^{20}$$`
$$10^{20}$$
- `$$10^{-2}$$`
$$10^{-2}$$
- `$$x^a_b$$`
$$x^a_b$$
- `$$x^{y^z}$$`
$$x^{y^z}$$

- `$\sum_{x=0}^n x^3$`

$$\sum_{x=0}^n x^3$$

- `$$\sum_{x=0}^n x^3$$`

$$\sum_{x=0}^{n} x^3$$

- `$$\int_0^n f(x)$$`

$$\int_0^n f(x)$$

- `$\lim_{n\to 0} {1 \over n} = \infty$`

$$\lim_{n\to 0} \frac{1}{n} = \infty$$

- `$$\lim_{n\to 0} {1 \over n} = \infty$$`

$$\lim_{n\to 0} \frac{1}{n} = \infty$$

Per ottenere una radice, LaTeX utilizza il comando '**\sqrt**', che però si comporta in modo leggermente diverso da TeX:

Comando	Descrizione
`\sqrt[`*n*`]{`*argomento*`}`	produce la radice ***n*** dell'argomento indicato.

Seguono alcuni esempi:

- `$$\sqrt{x^2+y^2}$$`

$$\sqrt{x^2+y^2}$$

- `$$\sqrt[3]{8}$$`

$$\sqrt[3]{8}$$

- `$$\sqrt{x+y \over z}$$`

$$\sqrt{\frac{x+y}{z}}$$

- `$$x^{\sqrt{y}}$$`

$$x^{\sqrt{y}}$$

LaTeX mette a disposizione gli stessi comandi di TeX per la scrittura dei nomi delle funzioni (sezione 49.17.8). Tuttavia, se si desidera scrivere il nome di una funzione in modo differente, lo si può inserire in una scatola orizzontale:

```
$$\mbox{sen}(2x)$$
```

$$\mbox{sen}(2x)$$

Per sovrapporre qualcosa si può usare il comando '**\stackrel**':

```
$$\stackrel{a}{B}$$
```

$$\stackrel{a}{B}$$

50.13.7 Matrici e sistemi di equazioni

Con LaTeX si rappresentano le matrici in modo molto simile alle tabelle:

```
\begin{array}{definizione_delle_colonne}
testo & testo [& testo]... \\
[testo & testo [& testo]... \\]
...
\end{array}
```


La definizione delle colonne avviene, come per l'ambiente '**tabular**', attraverso una stringa composta da tre lettere possibili:

Lettera	Significato mnemonico	Descrizione
`l`	*left*	rappresenta una colonna il cui contenuto deve essere allineato a sinistra;
`r`	*right*	rappresenta una colonna il cui contenuto deve essere allineato a destra;
`c`	*center*	rappresenta una colonna il cui contenuto deve essere allineato al centro.

Comunque, come nell'ambiente '**tabular**', è possibile anche inserire il simbolo '**|**' per ottenere una linea verticale e si può usare il comando '**\hline**' per inserire una linea orizzontale.

Segue un esempio molto semplice:

```
$$\left (
\begin{array}{cc|c}
a & b & c \\
d & e+f & g \\
\hline
h & i & j \\
\end{array}
\right )$$
```

Come si può osservare, l'ambiente '**array**' crea la tabella in cui vanno inseriti gli elementi della matrice, mentre la delimitazione esterna deve essere aggiunta espressamente:

$$\left (
\begin{array}{cc|c}
a & b & c \\
d & e+f & g \\
\hline
h & i & j
\end{array}
\right)$$

Trattandosi di una matrice, l'ambiente '**array**' si presta in pratica anche per realizzare un sistema di equazioni; è sufficiente regolare correttamente l'allineamento delle colonne, come nell'esempio seguente:

```
$$|x| =
\bigg \{
\begin{array}{rl}
x & x \geq 0 \\
-x & x \leq 0 \\
\end{array}
$$
```

$$|x| =
\bigg \{
\begin{array}{rl}
x & x \geq 0 \\
-x & x \leq 0 \\
\end{array}$$

L'ambiente '**array**' può essere usato in pratica nello stesso modo di '**tabular**', consentendo così la realizzazione di schemi più complessi di quanto mostrato qui. Eventualmente si consulti la sezione 50.11.2.

Tuttavia, esiste un ambiente speciale per elencare delle equazioni allineate a cui poter fare riferimento; si tratta di '**eqnarray**':

```
\begin{eqnarray}
testo & testo & testo \\
testo & testo & testo \\
...
\end{eqnarray}
```

In pratica, si tratta di una tabella in cui si presuppone la presenza di tre colonne, allineate rispettivamente a sinistra, al centro e a sinistra. Sarebbe semplicissimo realizzare questo con l'ambiente '**array**', ma la differenza sta nel fatto che ogni riga viene numerata in qualità di equazione. Si osservi l'esempio, in cui si inseriscono anche delle etichette che poi consentono di farvi riferimento:

```
\begin{eqnarray}
x & = & 2y \label{equazione_xy} \\
y & = & 3z \label{equazione_yz}
\end{eqnarray}

L'equazione \ref{equazione_xy} dice che... mentre
l'equazione \ref{equazione_yz} fa riferimento a...
```

Si osservi il fatto che l'ultima riga della matrice non è terminata con il simbolo '`\\`', perché diversamente si otterrebbe un'equazione in più, numerata, anche se invisibile:

$$x = 2y \quad (1)$$
$$y = 3z \quad (2)$$

L'equazione 1 dice che... mentre l'equazione 2 fa riferimento a...

Ancora più importante è notare che l'ambiente '**`eqnarray`**' viene dichiarato in un contesto normale, ma delimita un ambiente matematico particolare.

Eventualmente, per evitare di numerare una certa riga, si può usare il comando '**`\nonumber`**'; inoltre, se si aggiunge un asterisco al nome dell'ambiente, non viene mostrata la numerazione di alcuna equazione:

```
\begin{eqnarray*}
x & = & 2y \label{equazione_xy} \\
y & = & 3z \label{equazione_yz}
\end{eqnarray*}

L'equazione \ref{equazione_xy} dice che... mentre l'equazione
\ref{equazione_yz} fa riferimento a...
```

$$x = 2y$$
$$y = 3z$$

L'equazione 1 dice che... mentre l'equazione 1 fa riferimento a...

50.13.8 Teoremi, corollari e dichiarazioni simili

LaTeX ha una gestione sofisticata delle dichiarazioni riconducibili a teoremi, corollari e simili. Si parte dalla dichiarazione di un ambiente, al quale si associa un contatore, in modo tale da non porre limitazioni al genere di dichiarazioni che si vogliono gestire.

Inizialmente si definisce il tipo di contesto a cui appartiene un certo gruppo di dichiarazioni:

```
\newtheorem{ambiente}{Titolo}[contatore_di_riferimento]
```

```
\newtheorem{ambiente}[ambiente_collegato]{Titolo}
```

La seconda delle due forme permette di definire un ambiente la cui numerazione deve seguire quella di un altro già definito. In questo senso, nel campo indicato come ***ambiente_collegato***, va messo il nome di quello già esistente.

Per cominciare, si osservi l'esempio seguente, senza l'indicazione di alcun contatore di riferimento:

```
\newtheorem{teorema}{Teorema}
```

In questo modo si dichiara l'ambiente '**`teorema`**', con il titolo «Teorema», assieme al contatore '**`teorema`**' (a cui si accede facilmente con il comando '**`\thetheorem`**', creato implicitamente), usato in modo trasparente.

Nel dichiarare un ambiente di questo tipo, occorre evitare di scegliere nomi già usati da LaTeX.

Dopo la definizione dell'ambiente '**`teorema`**', si può usare in pratica questo ambiente; per esempio così:


```
\begin{teorema}
In teoria, non c'è differenza tra pratica e teoria.
\end{teorema}

\begin{teorema}
In pratica, la teoria è diversa dalla pratica.
\end{teorema}
```

Dal risultato della composizione si può osservare che il titolo, ovvero la parola «Teorema», viene messo in evidenza con un carattere nero, assieme al numero del contatore; subito dopo appare il testo del teorema in un carattere corsivo:

Teorema 1 *In teoria, non c'è differenza tra pratica e teoria.*

Teorema 2 *In pratica, la teoria è diversa dalla pratica.*

Nella definizione dell'ambiente '**`teorema`**' è stato omesso il contatore di riferimento, pertanto il conteggio dei teoremi parte da uno e prosegue di conseguenza. Si osservi invece l'esempio seguente, che presuppone un documento di tipo '**`article`**':

```
\newtheorem{teorema}{Teorema}[section]
```

In questo modo, si ottiene una numerazione subordinata al contatore '**`section`**', ovvero alla sequenza delle sezioni:

```
\section{Alcuni teoremi interessanti}
\begin{teorema}
In teoria, non c'è differenza tra pratica e teoria.
\end{teorema}

\begin{teorema}
In pratica, la teoria è diversa dalla pratica.
\end{teorema}

\section{Un'altra sezione}
\begin{teorema}
Un altro teorema.
\end{teorema}
```

Si può osservare che il numero mostrato dopo il titolo è strutturato, essendo composto da un prefisso che contiene il numero della sezione e da un suffisso che costituisce il numero relativo all'ambito della sezione:

1 Alcuni teoremi interessanti

Teorema 1.1 *In teoria, non c'è differenza tra pratica e teoria.*

Teorema 1.2 *In pratica, la teoria è diversa dalla pratica.*

2 Un'altra sezione

Teorema 2.1 *Un altro teorema.*

Con questa tecnica della dipendenza da un altro contatore, si possono costruire dei corollari la cui numerazione dipende dal teorema appena enunciato. Nell'esempio seguente si mostra anche la definizione degli ambienti usati:

```
\newtheorem{teorema}{Teorema}[section]
\newtheorem{corollario}{Corollario}[teorema]

\section{Alcuni teoremi e corollari interessanti}

\begin{teorema}
In teoria, non c'è differenza tra pratica e teoria.
\end{teorema}

\begin{corollario}
Bla bla bla bla.
\end{corollario}

\begin{teorema}
In pratica, la teoria è diversa dalla pratica.
\end{teorema}

\begin{corollario}
Bla bla bla bla.
\end{corollario}
```

1 Alcuni teoremi e corollari interessanti

Teorema 1.1 *In teoria, non c'è differenza tra pratica e teoria.*

Corollario 1.1.1 *Bla bla bla bla.*

Teorema 1.2 *In pratica, la teoria è diversa dalla pratica.*

Corollario 1.2.1 *Bla bla bla bla.*

Volendo invece usare una numerazione unica, sia per i teoremi, sia per i corollari, si può usare la seconda forma per la definizione dell'ambiente:

```
\newtheorem{teorema}{Teorema}[section]
\newtheorem{corollario}[teorema]{Corollario}
…
```

Ecco il risultato seguendo l'ultimo esempio già mostrato:

1 Alcuni teoremi e corollari interessanti

Teorema 1.1 *In teoria, non c'è differenza tra pratica e teoria.*

Corollario 1.2 *Bla bla bla bla.*

Teorema 1.3 *In pratica, la teoria è diversa dalla pratica.*

Corollario 1.4 *Bla bla bla bla.*

Naturalmente si possono definire anche ambienti di tipo diverso, a seconda delle esigenze; quello che conta è non usare nomi che possono creare conflitti per LaTeX.

Dal momento che queste definizioni sono numerate, diventa importante potervi fare riferimento. Per questo è necessario dichiarare un'etichetta con il comando '**`\label`**' all'interno dell'ambiente:

```
\newtheorem{teorema}{Teorema}[section]

\section{Alcuni teoremi e corollari interessanti}

\begin{teorema} \label{teorema-della-teoria}
In teoria, non c'è differenza tra pratica e teoria.
\end{teorema}

Purtroppo, il teorema \ref{teorema-della-teoria} viene
confutato dal teorema \ref{teorema-della-pratica}:

\begin{teorema} \label{teorema-della-pratica}
In pratica, la teoria è diversa dalla pratica.
\end{teorema}
```

1 Alcuni teoremi e corollari interessanti

Teorema 1.1 *In teoria, non c'è differenza tra pratica e teoria.*

Purtroppo, il teorema 1.1 viene confutato dal teorema 1.2:

Teorema 1.2 *In pratica, la teoria è diversa dalla pratica.*

50.14 File esterni

LaTeX prevede alcuni comandi per l'inclusione di file esterni; inoltre, è possibile creare file con informazioni ottenute in fase di composizione, in modo da poterle rielaborare successivamente con altri programmi.

50.14.1 Importazione

Esistono due comandi molto simili per dire a LaTeX di incorporare un file esterno:

Comando	Descrizione
`\input{`*nome_file*`}`	include il file nel punto in cui appare il comando (può essere usato anche nel preambolo);
`\include{`*nome_file*`}`	include il file nel punto in cui appare il comando, dopo un salto pagina (non può essere usato nel preambolo);

Comando	Descrizione
`\includeonly{`*nome_file_1*`,` *nome_file_2*`, …,`↩ *nome_file_n*`}`	usato nel preambolo, consente di selezionare quali file possono essere inclusi effettivamente con il comando '**`\include`**'.

Se i nomi dei file non contengono l'estensione, si intende che questa sia '`.tex`'; inoltre, come si può intuire dal nome, il comando '**`\includeonly`**' può interferire solo con le inclusioni attraverso '**`\include`**'.

50.14.2 Importazione letterale

Se si utilizza il pacchetto '**`verbatim`**' viene messo a disposizione un comando particolare per l'inserimento di file esterni da rappresentare nella composizione in modo letterale:

Comando	Descrizione
`\usepackage{verbatim}`	include il pacchetto '**`verbatim`**' (nel preambolo);
`\verbatiminput{`*nome_file*`}`	include il file nel punto in cui appare, rappresentandolo tale e quale.

50.14.3 Creazione di file

Per creare un file si deve prima definire il flusso in uscita, attribuendogli un nome, quindi questo deve essere aperto, inserendovi dei dati, infine lo si chiude. Di solito si creano file con una radice uguale a quella del file contenente il sorgente, attraverso il comando '**`\jobname`**':

Comando	Descrizione
`\newwrite\`*nome_flusso*	dichiara il nome del flusso da aprire in scrittura;
`\immediate\openout\`*nome_flusso*↩ ↪`\jobname.`*estensione*	crea il file con estensione '*.estensione*' e lo associa al flusso indicato;
`\immediate\write\`*nome_flusso*`{`*testo*`}`	aggiunge in coda al file una riga di testo, come indicato nell'argomento;
`\immediate\closeout\`*nome_flusso*	chiude il file corrispondente al flusso indicato;
`\jobname`	si espande nella stringa corrispondente alla radice del nome del file LaTeX sorgente.

L'esempio seguente mostra in che modo si potrebbe realizzare un indice generale elementare:

```
…
\chapter*{Indice generale}
\input{\jobname.indice}
…
\newwrite\miofile
\immediate\openout\miofile\jobname.indice
…
\chapter*{Introduzione}
\immediate\write\miofile{Introduzione {\arabic{page}}\par}
…
\chapter{Bla bla bla}
\immediate\write\miofile{Bla bla bla {\arabic{page}}\par}
…
\appendix
\chapter{Appendice}
\immediate\write\miofile{Appendice {\arabic{page}}\par}
…
\immediate\closeout\miofile
```

All'inizio dell'esempio si vede l'inclusione di un file con lo stesso nome del sorgente LaTeX ed estensione '`.indice`'. Come si può intendere, la prima volta questo file non esiste e l'inclusione non produce alcun risultato. Successivamente si crea questo file, quindi,

all'inizio di ogni sezione principale si annota il titolo di questa e la pagina in cui si trova. Nella composizione successiva, il file viene incluso e successivamente rigenerato, probabilmente con numeri di pagina diversi, a causa dell'allungamento del documento; nella terza composizione si dovrebbe ottenere un risultato soddisfacente. Ecco come si potrebbe presentare il file generato in questo modo:

```
Introduzione {1}\par
Bla bla bla {10}\par
Appendice {25}\par
```

Nell'esempio si deve osservare che '**\arabic{page}**' si è trasformato correttamente nel numero della pagina e '**\par**' è rimasto intatto. Nel caso si inserissero delle macro, si potrebbe ottenere un risultato indesiderabile. Per esempio, si potrebbe voler separare il titolo dal numero della pagina con dei puntini di sospensione elastici:

```
\immediate\write\miofile{Introduzione\dotfill{\arabic{page}}\par}
```

Il file generato si presenterebbe così:

```
Introduzione\unhbox \voidb@x \cleaders ↩
↪\hbox to.44em{\hss .\hss }\hfill \kern \z@ {1}\par
```

In questi casi si può usare il comando '**\string**' per evitare che la barra obliqua inversa del comando '**\dotfill**' venga presa in considerazione:

```
\immediate\write\miofile{Introduzione\string\dotfill{\arabic{page}}\par}
```

50.15 Riferimenti

- *General documentation about (La)TeX*, *http://tex.loria.fr/english/general.html*
- Tobias Oetiker, *The not so short introduction to LaTeX 2e*, 2001, *http://tobi.oetiker.ch/lshort/*
- Leslie Lamport et al., *LaTeX 2e*, *http://www.emerson.emory.edu/services/latex/latex2e/latex2e_toc.html* , *info latex*
- Michel Goossens, Franck Mittelbach, Alexander Samarin, *The LaTeX Companion*, Addison-Wesley, 1994
- Leslie Lamport, *LaTeX, A document preparation system, User's guide and reference manual*, Addison-Wesley, seconda edizione, 1994
- Claudio Beccari, *LaTeX, Guida a un sistema di editoria elettronica*, Hoepli, 1991, ISBN 88-203-1931-4
- Robert Kiesling, *teTeX HOWTO: The Linux-teTeX Local Guide*, *http://tldp.org/HOWTO*
- Documentazione interna della distribuzione teTeX: '`texmf/doc/*`', in particolare '`texmf/doc/latex/latex2e-html/`', dove si trova una guida di LaTeX in HTML
- Documentazione interna Info: *info web2c*, *info tex*, *info latex*, *info pdftex*
- Sebastian Rahtz, Hàn Thé Thành, *The pdfTeX user manual*, '`texmf/doc/pdftex/manual.pdf`'
- Vidar Bronken Gundersen, Rune Mathisen, *SGML/XML character entity reference*, *http://www.bitjungle.com/isoent/*

[1] Tutti i comandi che definiscono un livello di suddivisione del documento, da '**\part**' a '**\subparagraph**', consentono di utilizzare un asterisco alla fine del nome (come per esempio '**\section*{**…**}**'); l'utilizzo di questo porta a sopprimere la numerazione della voce e il suo inserimento nell'indice generale (ammesso che queste due cose siano previste normalmente).

[2] Eventualmente, per ottenere il carattere «tilde», oltre a usare il comando '**\textasciitilde**' si può usare invece '**$\sim$**', per produrre un carattere che a volte viene ritenuto più grazioso.

[3] Naturalmente, '**makeindex**' può essere usato anche specificando delle opzioni; in tal caso si può consultare la pagina di manuale *makeindex(1)*.

[4] È bene ricordare che la tabella si compone di quattro colonne, dove la seconda e la terza non sono separate da una linea verticale.

[5] Qui non sono state usate della parentesi quadre sintattiche per non creare confusione.

SGML

L'SGML non è un «linguaggio di scrittura» da imparare e usare così com'è. Al contrario, è un linguaggio per definire il modo in cui il testo deve essere scritto: solo dopo si può iniziare a scrivere secondo le regole stabilite.

Volendo fare un abbinamento con i linguaggi di programmazione, sarebbe come se prima si dovesse definire il linguaggio stesso, per poi poter scrivere i programmi secondo quelle regole.

La descrizione che viene fatta potrebbe risultare noiosa, considerato che solo dopo molte sezioni si mostra in che modo realizzare effettivamente il proprio DTD e applicarlo a un documento. Considerata la complessità dei concetti espressi, si ritiene più conveniente una spiegazione che parte dal basso, piuttosto che usare un approccio inverso, il quale presumerebbe una conoscenza minima di partenza.

51.1 DTD: definizione del tipo di documento

Le regole che definiscono la struttura e la scomposizione del documento, assieme a quasi tutte le altre che governano la logica dell'SGML, sono contenute nel DTD.

Queste regole possono essere permissive o restrittive, in funzione degli obiettivi che ci si prefigge; ovvero, in funzione del contenuto di quel tipo di documento e delle cose che con questo ci si aspetta di fare.

La complessità del mondo reale, fa sì che non ci sia modo di realizzare un DTD unico che vada bene per tutti gli scopi. Un DTD

ipotetico, che volesse andare bene un po' per tutto, dovrebbe essere anche qualcosa di estremamente generico e permissivo, annullando tutti i benefici dell'utilizzo dell'SGML.

Esempi reali di DTD «tuttofare» sono quelli delle prime versioni dell'HTML, in cui tutto si concentra nella definizione di elementi il cui scopo prevalente è definire, anche se solo vagamente, l'aspetto finale che dovrebbe avere il risultato. Lo scopo dell'SGML non è quello di stabilire il risultato finale del documento, tuttavia, si può benissimo predisporre un DTD orientato a questo obiettivo. Ma questo, nel caso dell'HTML, giustifica poi l'estrema debolezza della sua struttura, dove è ammesso quasi tutto.

È difficile comprendere subito il significato pratico di questo approccio: la definizione del tipo di documento e poi la scrittura del testo. Lo si può comprendere solo quando si lavora assiduamente nell'ambito della produzione di documentazione, quando ci si accorge che le proprie esigenze sono diverse da quelle degli altri, per cui diventa difficile adattarsi all'uso di modelli già esistenti.

51.1.1 Elementi

Dal punto di vista di SGML, una singola unità di testo la cui dimensione varia a seconda del contesto, è un ***elemento***, a cui si impone l'attribuzione di un nome. SGML non fornisce alcun modo per attribuire un significato agli elementi del testo, tranne per il fatto di avergli dato un nome. Piuttosto, attraverso un analizzatore SGML, è possibile verificare che questi siano collocati correttamente secondo le relazioni stabilite.

I nomi degli elementi, sono definiti tecnicamente ***identificatori generici***, utilizzando la sigla GI (*Generic identifier*).

Nel sorgente SGML, gli elementi sono indicati normalmente attraverso l'uso di marcatori che hanno la forma consueta '**<**...**>**' e '**</**...**>**', dove il primo inizia l'elemento nominato tra le parentesi angolari e il secondo chiude l'elemento. Per esempio, si potrebbe definire l'elemento '**acronimo**' e utilizzarlo nel testo nel modo seguente:

```
...Il gruppo <acronimo>ILDP</acronimo> si occupa di...
```

Il significato che questo elemento può avere, non è definito dall'SGML. Il fatto di avere delimitato l'elemento '**acronimo**' potrebbe servire per estrarre dal documento tutte le sigle utilizzate, per inserire queste in un indice particolare, oppure solo per fini stilistici di evidenziamento uniforme.

La difficoltà nella scrittura di un testo in SGML si riduce a questo: utilizzare i marcatori necessari a identificare correttamente i vari elementi del testo, secondo le regole stabilite nella definizione del documento stesso (il DTD).

51.1.1.1 Abbreviazioni nell'indicazione degli elementi

Prima ancora di iniziare a vedere il contenuto del DTD, è bene chiarire che esistono altri modi per delimitare un elemento SGML. Per la precisione, si tratta di abbreviazioni di cui alcuni autori non riescono a fare a meno. La scrittura di un sorgente SGML è un po' come quella di un sorgente di un linguaggio di programmazione: si può essere concisi o prolissi. Di solito, quando si è concisi si scrive del codice difficile da leggere, mentre in generale è meglio scrivere tutto in forma chiara senza risparmiare. L'esempio visto in precedenza,

```
...Il gruppo <acronimo>ILDP</acronimo> si occupa di...
```

può essere abbreviato in

```
...Il gruppo <acronimo>ILDP</> si occupa di...
```

e anche nel modo seguente, che però porta con sé un vincolo importante: non si possono usare delle barre oblique all'interno dell'elemento abbreviato in questo modo.

```
...Il gruppo <acronimo/ILDP/ si occupa di...
```

Con questi sistemi, oltre a rendere il sorgente SGML poco leggibile, si rischia di non ottenere i risultati che si attendono se gli strumenti di elaborazione utilizzati non riconoscono tali estensioni del linguaggio.

51.1.1.2 Primo impatto con un DTD

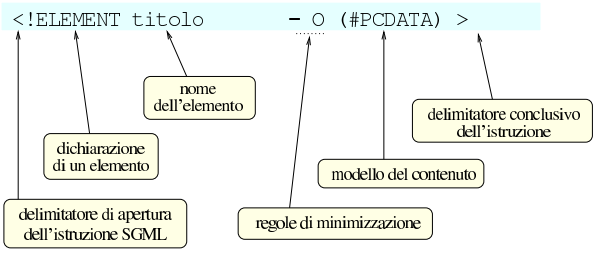
La definizione del DTD è ottenuta da una serie di istruzioni dichiarative composte secondo una sintassi molto semplice. L'esempio seguente rappresenta le istruzioni necessarie a definire gli elementi di un tipo di documento ipotetico definibile come '**relazione**'.

```
<!ELEMENT relazione     - - (titolo?, data, contenuto) >
<!ELEMENT titolo        - o (#PCDATA) >
<!ELEMENT data          - o (#PCDATA) >
<!ELEMENT contenuto     - o (paragrafo+, firma+) >
<!ELEMENT paragrafo     - o (#PCDATA) >
<!ELEMENT firma         - o (#PCDATA) >
```

Ognuna delle righe che appaiono nell'esempio rappresenta una dichiarazione di un elemento SGML. Una dichiarazione, di qualunque tipo, è delimitata da una parentesi angolare aperta (il simbolo di minore), seguita immediatamente da un punto esclamativo ('**<!**') e da una parentesi angolare chiusa ('**>**').

La dichiarazione di un elemento si compone poi della parola chiave '**ELEMENT**', seguita dal nome dell'elemento, dalle regole di minimizzazione rappresentate da due caratteri e da un modello del contenuto.

Figura 51.6. Scomposizione delle varie parti della dichiarazione di un elemento SGML.



Le varie parti che compongono qualunque tipo di dichiarazione SGML sono separate da spazi orizzontali (caratteri spazio, o tabulazioni orizzontali) oppure anche da interruzioni di riga, permettendo così di proseguire le istruzioni su più righe distinte.

51.1.2 Regole di minimizzazione

Le ***regole di minimizzazione***, rappresentate da due caratteri staccati, indicano l'obbligatorietà o meno dell'utilizzo del marcatore di apertura e di chiusura per l'elemento dichiarato. Il primo dei due simboli rappresenta l'apertura, il secondo la chiusura. Un trattino indica che il marcatore è obbligatorio, mentre la lettera '**O**' sta per «opzionale» e indica così che può essere omesso:

Regole di minimizzazione	Descrizione
- -	sono obbligatori entrambi i marcatori;
- O	è obbligatorio il marcatore iniziale, mentre quello finale è facoltativo;
O -	il marcatore iniziale è facoltativo, mentre quello finale è obbligatorio (di solito non capita questa situazione);
O O	sono facoltativi entrambi i marcatori.

Nell'esempio mostrato in precedenza, solo l'elemento '**relazione**' richiede l'utilizzo di marcatori di apertura e di chiusura, mentre tutti gli altri possono essere indicati utilizzano il solo marcatore di apertura. In pratica, il contesto permette di individuare dove finiscano tali elementi nel testo.

La possibilità o meno di rendere facoltativo l'uso dei marcatori di apertura e di chiusura non è solo un fatto di gusto, in quanto dipende anche dall'organizzazione del tipo di documento. Se le dichiarazioni diventano ambigue, non si possono più distinguere gli elementi nel testo SGML.

51.1.3 Modello del contenuto

La parte finale della dichiarazione di un elemento SGML è il ***modello del contenuto***, che si distingue perché è racchiuso tra parentesi tonde. Serve a descrivere il tipo di contenuto che può avere l'elemento e si può esprimere attraverso parole riservate che hanno un significato preciso, come '**#PCDATA**' (*Parsed character data*) che rappresenta una qualunque sequenza di caratteri valida, oppure attraverso l'indicazione di nomi di altri elementi che possono (o devono) essere contenuti in qualche modo.

Il modello del contenuto, può articolarsi in modo molto complesso, allo scopo di definire le relazioni tra gli elementi contenuti.

Per il momento, è bene osservare che un elemento, il cui modello del contenuto sia composto esclusivamente della parola riservata '**#PCDATA**', non può contenere al suo interno altri tipi di elementi. Il significato di alcune delle parole riservate più comuni, utilizzabili per definire il contenuto di un elemento, sono riportate più avanti in questo capitolo, dopo la presentazione di altri concetti essenziali, necessari per comprenderne il senso.

51.1.3.1 Indicatori di ripetizione

Il modello del contenuto utilizza un sistema abbastanza complesso per definire la possibilità di contenere più elementi dello stesso tipo e per indicare raggruppamenti di elementi. Per indicare la ripetizione, viene usato un simbolo alla fine dell'oggetto a cui si riferisce, chiamato ***indicatore di ripetizione*** (*occurrence indicator*):

Indicatore di ripetizione	Descrizione
+	il segno '**+**' usato come suffisso, rappresenta una o più ripetizioni dell'elemento;
?	il segno '**?**' usato come suffisso, rappresenta zero o al massimo un'occorrenza dell'elemento;
*	il segno '*****' usato come suffisso, rappresenta zero o più ripetizioni dell'elemento;
	se non viene usato nessun suffisso, l'elemento indicato deve essere usato esattamente una volta.

Dall'esempio mostrato in precedenza viene ripreso l'estratto seguente, nel quale si può osservare che: l'elemento '**titolo**' può apparire al massimo una volta all'interno di '**relazione**' (precisamente all'inizio di questo elemento); l'elemento '**paragrafo**' deve essere contenuto almeno una volta all'interno dell'elemento '**contenuto**' (lo stesso vale per l'elemento '**firma**'); l'elemento '**firma**' può contenere solo caratteri normali senza altri elementi.

```
<!ELEMENT relazione     - - (titolo?, data, contenuto) >
<!ELEMENT contenuto     - O (paragrafo+, firma+) >
<!ELEMENT firma         - O (#PCDATA) >
```

51.1.3.2 Connettori

Quando un elemento deve poter contenere diversi tipi di elementi, è necessario usare dei simboli, detti ***connettori***, per stabilirne la relazione:

Connettore	Descrizione
,	la virgola ('**,**') indica che l'elemento precedente e quello successivo devono apparire nell'ordine in cui sono;
&	la e-commerciale ('**&**') indica che l'elemento precedente e quello successivo devono essere presenti entrambi, ma possono apparire in qualunque ordine;
|	la barra verticale ('**|**') indica che solo uno tra i due elementi che connette può apparire.

Riprendendo il solito estratto dell'esempio già mostrato precedentemente, si può osservare l'uso della virgola in qualità di connettore:


```
<!ELEMENT relazione     - - (titolo?, data, contenuto) >
<!ELEMENT contenuto     - O (paragrafo+, firma+) >
<!ELEMENT firma         - O (#PCDATA) >
```

L'elemento '**relazione**' può contenere al massimo un titolo all'inizio, quindi deve apparire un elemento '**data**' e dopo di questo anche un elemento '**contenuto**'. L'elemento '**contenuto**' deve contenere uno o più elementi '**paragrafo**' a partire dall'inizio, mentre in coda deve avere uno o più elementi '**firma**'.

```
<!ELEMENT nominativo    - - (nome & cognome) >
<!ELEMENT voce          - - (punto | numero) >
```

Per completare gli esempi sull'uso dei connettori, si osservi quanto mostrato sopra. L'elemento '**nominativo**' deve contenere un elemento '**nome**' e un elemento '**cognome**', in qualunque ordine; l'elemento '**voce**' può contenere solo un elemento a scelta tra '**punto**' e '**numero**'.

51.1.3.3 Raggruppamenti

All'interno di un modello di contenuto, è possibile indicare dei raggruppamenti che esprimono in pratica dei sottomodelli, a cui poter applicare gli indicatori di ripetizione e i connettori. Per questo si usano le parentesi tonde. Si osservi l'esempio seguente:

```
<!ELEMENT figure - - ( (eps | ph), img*, caption?) >
```

L'elemento '**figure**' deve contenere un'occorrenza del sottogruppo '**(eps | ph)**', zero o più ripetizioni dell'elemento '**img**' e al massimo un'occorrenza di '**caption**', nell'ordine descritto. Il sottogruppo '**(eps | ph)**' rappresenta una singola occorrenza di '**eps**' oppure '**ph**'.

Quando si utilizzano gli operatori di ripetizione assieme ai raggruppamenti, possono nascere degli equivoci. Ammesso che ciò possa avere senso, si osservi la variante seguente dell'esempio già presentato:

```
<!ELEMENT figure - - ( (eps | ph)+, img*, caption?) >
```

È stato aggiunto il segno '**+**' dopo il gruppo '**(eps | ph)**'. In questo modo, si intende che sia possibile l'inserimento iniziale di una serie indefinita di elementi '**eps**' o '**ph**', in qualunque ordine, purché ce ne sia almeno uno dei due. Quindi, non è necessario che si tratti solo di elementi '**eps**' o solo di '**ph**'.

51.1.3.4 Eccezione

Se nella definizione di un elemento si vogliono indicare delle eccezioni a quanto definito dal modello di contenuto, si può indicare un gruppo di elementi successivo al modello del contenuto.

Questo gruppo può essere preceduto dal segno '**+**' o dal segno '**-**' indicando rispettivamente un'eccezione di inclusione, o un'eccezione di esclusione.

```
<!ELEMENT address - O (#PCDATA) +(newline) >
```

L'esempio mostra l'elemento '**address**' contiene caratteri normali, ma che può includere eccezionalmente anche l'elemento '**newline**'.

```
<!ELEMENT acronimo - - (#PCDATA) -(acronimo) >
```

L'esempio mostra l"elemento '**acronimo**' contiene caratteri normali e che non può includere se stesso (a essere precisi, non è necessario dichiarare una cosa del genere, dal momento che il contenuto '**#PCDATA**' non ammette altri elementi al suo interno).[1]

51.1.3.5 Elementi vuoti

Alcuni tipi di elementi non sono fatti per circoscrivere una zona di testo, ma solo per rappresentare qualcosa che si trova in un certo punto. Questi elementi, non vengono dichiarati con un modello di contenuto tra parentesi, ma con l'utilizzo della parola chiave '**empty**'.

L'esempio seguente, dichiara l'elemento '**toc**' che non può contenere alcunché.

```
<!ELEMENT toc - O EMPTY>
```

Tipicamente, tali elementi, sono dichiarati in modo che il marcatore di chiusura sia solo facoltativo. Non potendo contenere alcunché, sarebbe perfettamente inutile renderlo obbligatorio.

51.1.4 Dichiarazione multipla

Eventualmente, un gruppo di elementi che abbiano le stesse caratteristiche, cioè le stesse regole di minimizzazione e lo stesso modello del contenuto, può essere dichiarato in una sola istruzione. L'esempio seguente dovrebbe essere sufficiente a comprendere il meccanismo.

```
<!ELEMENT (annotazione|avvertimento|pericolo ) - - (#PCDATA) >
```

51.1.5 Attributi

Un elemento può prevedere la presenza di uno o più attributi. Si tratta di informazioni che non compongono il contenuto dell'elemento, ma di qualcosa che, non potendo apparire nel testo, serve per qualche ragione ai programmi che elaborano successivamente il documento. Il classico esempio è costituito da quei marcatori utilizzati per i riferimenti incrociati. L'esempio seguente mostra l'uso di un elemento vuoto, denominato '**ref**', contenente l'attributo '**point**' a cui viene dato il valore '**esempio**':

```
Si veda il capitolo <ref point="esempio"> che contiene
molti esempi utili al riguardo.
```

È importante osservare che il valore assegnato a un attributo deve essere delimitato attraverso apici doppi (come mostrato nell'esempio), oppure attraverso apici singoli. Eccezionalmente, è possibile assegnare un valore senza alcuna delimitazione, quando si tratta di una sola parola composta da lettere alfabetiche, cifre numeriche, trattino normale ('**-**'), trattino basso ('_'), due punti ('**:**').

L'esempio seguente mostra la dichiarazione dell'elemento '**ref**', già presentato nell'esempio, tenendo conto che il suo scopo è quello di essere utilizzato come riferimento a una parte del documento identificata attraverso il valore assegnato all'attributo '**point**'.

```
<!ELEMENT ref - O EMPTY>
<!ATTLIST ref
        point IDREF #REQUIRED
        name CDATA "riferimento">
```

Attraverso l'istruzione '**ATTLIST**' si definiscono gli attributi di un elemento. Dopo l'indicazione del nome dell'elemento a cui si fa riferimento, segue l'elenco degli attributi, ognuno dei quali inizia con un codice di interruzione di riga seguito eventualmente da altri tipi di spazi. Ciò significa che l'istruzione '**ATTLIST**' deve essere composta proprio come indicato dall'esempio, solo i rientri sono facoltativi.

L'esempio indica che l'elemento '**ref**' contiene due attributi: '**point**' e '**name**'. Il primo è obbligatorio ('**#REQUIRED**'), mentre per il secondo è stato indicato un valore predefinito, nel caso non venga specificato espressamente ('**riferimento**').

Il tipo di contenuto di un attributo viene definito attraverso delle parole chiave, che possono essere indicate usando lettere maiuscole o minuscole indifferentemente. Di seguito ne vengono descritte alcune:

Parola chiave	Descrizione
CDATA	rappresenta una stringa di qualunque tipo di carattere, ammettendo anche simboli di punteggiatura o altro, che comunque mantiene solo il suo significato letterale (*Character data*);
NMTOKEN	rappresenta qualunque tipo di carattere alfanumerico (lettere, numeri e spazi soltanto), che dovrebbe comporre un nome (*Name token*);
NUMBER	rappresenta solo cifre numeriche, cioè un numero;
ID	rappresenta un identificatore unico per quel tipo di documento, costituito da un nome senza caratteri speciali o segni di punteggiatura, che viene utilizzato successivamente per farvi riferimento;

Parola chiave	Descrizione
IDREF	indica che l'attributo deve essere un puntatore valido a un identificatore di un attributo '**ID**', corrispondente in un altro elemento.

È importante osservare che la parola chiave '**CDATA**' viene usata anche in altre situazioni con un significato simile, ma non identico. Nel caso definisca il contenuto di un attributo, è ammesso l'uso di macro (entità) che vengono espanse.

Il tipo di contenuto di un attributo, può essere indicato in modo preciso attraverso una serie di scelte alternative. In tal caso, invece di utilizzare le parole chiave già elencate, si indicano le stringhe alternative, separate dalla barra verticale, tra parentesi tonde. Per esempio, '**(bozza | finale)**' rappresenta la possibile scelta tra le due parole '**bozza**' e '**finale**'.

L'ultimo dato da inserire per ogni attributo è il valore predefinito, oppure una parola chiave a scelta tra le seguenti:

Parola chiave	Descrizione
#REQUIRED	rappresenta l'obbligatorietà dell'inserimento del valore;
#IMPLIED	rappresenta un attributo facoltativo;
#CURRENT	in mancanza di un'indicazione esplicita, rappresenta l'utilizzo dell'ultimo valore assegnato allo stesso attributo dello stesso elemento;
#FIXED	rappresenta un valore predefinito e non modificabile, che, se usato, deve avere il valore stabilito.

Nel caso particolare dell'attributo definito con la parola chiave '**#FIXED**', a questa segue necessariamente la stringa fissata.

Tra tutti, merita attenzione la coppia '**ID**' e '**IDREF**'. Questi tipi di attributi possono essere molto utili per definire dei riferimenti incrociati all'interno del documento, quando la loro validità deve essere controllata con gli strumenti di convalida SGML. Si osservi l'esempio seguente:

```
<!ELEMENT label - O EMPTY>
<!ATTLIST label
        identity ID #REQUIRED>

<!ELEMENT ref - O EMPTY>
<!ATTLIST ref
        point IDREF #REQUIRED>
```

Nell'esempio si mostra la dichiarazione di un elemento '**label**' che non può contenere testo, in quanto serve solo per definire l'attributo '**identity**', di tipo '**ID**'. Questo permette l'utilizzo di marcatori simili a '**<label identity="miaetichetta">**', dove viene assegnato all'attributo '**identity**' un nome sempre diverso, allo scopo di identificare qualcosa. Sotto, la dichiarazione dell'elemento '**ref**' mostra un altro elemento che non può contenere testo, ma solo un attributo denominato '**point**', di tipo '**IDREF**', che può quindi contenere solo il nome di un identificatore già usato in un altro elemento con l'attributo '**ID**'.

In pratica, se nel testo SGML si dovesse utilizzare da qualche parte il marcatore '**<label identity="miaetichetta">**', in un altro punto sarebbe valido il marcatore '**<ref point="miaetichetta">**', perché l'identificatore '**miaetichetta**' esiste effettivamente.

Ricapitolando, un attributo '**ID**' di un marcatore è valido quando è unico nel documento SGML che si scrive, mentre un attributo '**IDREF**' è valido quando esiste il valore corrispondente di un attributo '**ID**'.

Spesso, per cose del genere, si preferisce usare attributi di tipo '**CDATA**', per permettere l'utilizzo di caratteri di ogni tipo, togliendo però all'SGML la possibilità di controllare la validità di tali riferimenti incrociati.

51.1.6 Entità

Con questo termine, ***entità***, si fa riferimento a due tipi di oggetti: macro per la sostituzione di stringhe (entità generali) o macro per la sostituzione di nomi all'interno di istruzioni SGML (entità parametriche).

Le macro per la sostituzione di stringhe, una volta dichiarate, si utilizzano all'interno del sorgente SGML come abbreviazioni o come un modo per identificare lettere o simboli che non possono essere usati altrimenti. Per esempio, utilizzando le entità ISO 8879:1986, la frase «Wer bekommt das größte Stück Torte?» può essere scritta nel sorgente nel modo seguente:

```
Wer bekommt das gr&ouml;&szlig;te St&uuml;ck Torte?
```

Le entità generali, quindi, sono identificate nel testo SGML perché iniziano con la e-commerciale ('**&**') e terminano con un punto e virgola. È bene osservare che il punto e virgola non è obbligatorio in ogni situazione, ma solo quando il carattere successivo sia diverso da uno spazio orizzontale o da un codice di interruzione di riga. In generale, però, sarebbe bene usare sempre il punto e virgola. La tabella 51.25 elenca alcune macro delle entità standard più importanti.

Tabella 51.25. Alcune macro delle entità standard secondo le specifiche ISO 8879:1986.

á	á	Á	Á	ö	ö	Ö	Ö
â	â	Â	Â	ß	ß		
à	à	À	À	ú	ú	Ú	Ú
å	å	Å	Å	û	û	Û	Û
ã	ã	Ã	Ã	ù	ù	Ù	Ù
ä	ä	Ä	Ä	ü	ü	Ü	Ü
æ	æ	Æ	Æ	ý	ý	Ý	Ý
ç	ç	Ç	Ç	ÿ	ÿ		
é	é	É	É	&	&	@	@
ê	ê	Ê	Ê	*	*		
è	è	È	È	ˆ	^	˜	~
ë	ë	Ë	Ë	©	©		
í	í	Í	Í	$	$	%	%
î	î	Î	Î	#	#		
ì	ì	Ì	Ì	!	!	¡	¡
ï	ï	Ï	Ï	?	?	¿	¿
ñ	ñ	Ñ	Ñ	‐	-	_	_
ó	ó	Ó	Ó	\	\		
ô	ô	Ô	Ô	"	"		
ò	ò	Ò	Ò	<	<	>	>
ø	ø	Ø	Ø	[	[	]	]
õ	õ	Õ	Õ	{	{	}	}

Le entità standard ISO 8879, sono distinte in 19 gruppi, che in parte si sovrappongono (a volte si ripetono alcune dichiarazioni nello stesso modo). Questi 19 gruppi di entità corrispondono ad altrettanti file, per i quali esiste anche un nome stabilito.

L'altro tipo di macro, riguarda invece la sostituzione all'interno delle istruzioni SGML, cioè nella dichiarazione del DTD.

L'esempio seguente mostra la dichiarazione dell'elemento '**p**' che può contenere l'elemento o gli elementi indicati all'interno della macro '**%inline;**'.

```
<!ELEMENT p - O (%inline;) >
```

La dichiarazione di un'entità avviene utilizzando l'istruzione '**ENTITY**'. L'esempio seguente mostra la dichiarazione di un'entità da utilizzare nel sorgente SGML.

```
<!ENTITY agrave "\'a">
```

In questo caso, si vuole che la macro '**à**' venga sostituita con la stringa '**\'a**'. Evidentemente, questa trasformazione non ha niente a che vedere con SGML. È semplicemente una scelta motivata dal tipo di programma utilizzato successivamente per rielaborare il risultato generato dall'analizzatore SGML.

L'esempio seguente mostra la dichiarazione di due entità da utilizzare all'interno delle istruzioni SGML.

```
<!ENTITY % emph " em | concept | cparam ">
<!ENTITY % inline "(#PCDATA | %emph;)*">
```

La dichiarazione di questo tipo di entità si distingue perché viene utilizzato il simbolo di percentuale subito dopo la parola '**ENTITY**', staccandolo da questa e anche dal nome dell'entità successivo. Anche in questo caso si utilizza solo come pura sostituzione di stringhe, per cui la dichiarazione di '**%inline;**', facendo a sua volta riferimento a '**%emph;**', è equivalente a quella seguente:

```
<!ENTITY % inline "(#PCDATA | em | concept | cparam )*">
```

Naturalmente, una macro può contenere anche il riferimento a un'altra macro. Per esempio, la dichiarazione dell'ipotetico elemento '**p**', fatta nel modo seguente,

```
<!ELEMENT p - O (%inline;) >
```

è equivalente alla dichiarazione:

```
<!ELEMENT p - O ((#PCDATA | em | concept | cparam )*) >
```

51.1.6.1 Acquisizione dall'esterno

Le entità di qualunque tipo, possono essere dichiarate abbinando una stringa a una macro, come è stato mostrato in precedenza. In alternativa, a una macro si può abbinare un file esterno (file inteso nel senso più ampio possibile). In tal caso, si utilizza la parola chiave '**SYSTEM**' come nell'esempio seguente:

```
<!ENTITY capitolo2 SYSTEM "capitolo2.sgml">
```

In tal modo, quando nel documento SGML si utilizza la macro '**&capitolo2;**' e poi lo si elabora attraverso un analizzatore SGML, si ottiene l'inserimento del file '`capitolo2.sgml`'. Più o meno ciò che si fa normalmente con le direttive di un precompilatore di un linguaggio di programmazione.

Nello stesso modo si può fare per dichiarare un'entità parametrica, come nell'esempio seguente:

```
<!ENTITY % isoent SYSTEM "isoent.txt">
```

L'esempio mostra la dichiarazione della macro '**%isoent;**', riferita al file '`isoent.txt`'. Per utilizzare questa macro, bisogna sapere a cosa si riferisce; trattandosi di un file, è logico pensare che si tratti di un testo articolato su più righe, quindi inadatto all'inserzione all'interno delle istruzioni. Generalmente, una macro del genere serve a incorporare un pezzo di DTD dall'esterno.

```
%isoent;
```

Come si vede dall'esempio, è normale vedere la chiamata di una macro di questo tipo, da sola, all'esterno di qualunque istruzione del DTD. L'esempio mostrato è comunque significativo: rappresenta l'inclusione di un file che presumibilmente, dal nome, serve a incorporare le entità ISO, cioè quelle standard riferite alle lettere accentate e ai simboli speciali.

A questo proposito, potrebbero esistere diversi file, del tipo: ‘`isoent.latex.txt`’, ‘`isoent.html.txt`’,... che prima di avviare l’analizzatore SGML vengono sostituiti al file ‘`isoent.txt`’, in modo da ottenere la sostituzione corretta in base all’elaborazione successiva che si vuole ottenere (LaTeX, HTML, ecc.).

Se non fosse ancora chiaro, ecco come potrebbe essere composto l’ipotetico file ‘`isoent.txt`’ quando si vogliono le sostituzioni corrette per LaTeX.

```
<!ENTITY agrave "\‘a">
<!ENTITY Agrave "\‘A">
<!ENTITY egrave "\‘e">
<!ENTITY Egrave "\‘E">
<!ENTITY eacute "\’e">
<!ENTITY Eacute "\’E">
…
```

L’acquisizione di una macro da un file esterno può essere dichiarata senza specificare esplicitamente il file, lasciando che l’analizzatore trovi il file corretto in base a un catalogo SGML. L’argomento viene ripreso in seguito, comunque, in questo tipo di dichiarazione, manca il nome del file.

```
<!ENTITY capitolo2 SYSTEM>
```

```
<!ENTITY % isoent SYSTEM>
```

Solitamente, si preferisce includere in questo modo solo le macro parametriche, cosa che può essere compresa intuitivamente in seguito.

51.1.6.2 Codici macro speciali

È bene ribadire che l’uso delle entità standard (ISO), permette di rendere il testo SGML indipendente dalla piattaforma utilizzata. Tuttavia, la dichiarazione della sostituzione dipende dalla piattaforma e, come si è mostrato, si tendono a predisporre diversi schemi di sostituzione per le diverse piattaforme a cui si vuole fare riferimento.

In situazioni eccezionali, può essere conveniente indicare i caratteri per numero, decimale o esadecimale, attraverso una notazione simile a quella delle entità normali. Per esempio, se si usa la codifica ISO 8859-1 (Latin 1), la macro ‘**`&#232;`**’, oppure la macro ‘**`&#xe8;`**’ corrisponde alla lettera ‘**è**’ (la «e» accentata normale).

Questa possibilità è fondamentale proprio quando si definiscono le stringhe di sostituzione per una piattaforma determinata (hardware-software), in cui si debbano indicare caratteri speciali identificati dal numero corrispondente.

```
<!ENTITY egrave "&#232;">
```

Potrebbe sembrare che un testo SGML non possa utilizzare una codifica particolare, quale ISO 8859-1 o altro. Non è così. L’SGML mette a disposizione le entità standard, ma ciò non toglie che si possa decidere di usare comunque una codifica (ASCII) estesa come Latin 1 o altro. Ovviamente questo rende il testo dipendente dalla piattaforma, precisamente dalla codifica. Volendo essere precisi, la codifica utilizzabile dipende dalla dichiarazione SGML, cosa che viene descritta nella sezione 51.3.

51.1.7 Sezioni marcate

Le sezioni marcate sono una specialità di SGML, poco usata e poco conosciuta. Si tratta di istruzioni che vengono inserite nel testo SGML (non nel DTD) e servono a vario titolo per delimitare del testo per qualche scopo.

Una sezione marcata si compone di un sorta di marcatore di apertura e di una sorta di marcatore di chiusura. Il marcatore di apertura contiene una parola chiave che ne identifica il comportamento. Si osservi l’esempio seguente:

```
<![INCLUDE[
Questa parte del testo è inclusa nell’elaborazione SGML.
]]>
```


Come si può intuire, la sezione marcata dell’esempio è introdotta da ‘**`<![INCLUDE[`**’ ed è terminata da ‘**`]]>`**’. In questo caso, la parola chiave ‘**`INCLUDE`**’ indica che il testo contenuto nella sezione marcata deve essere incluso nell’elaborazione (anche se ciò, per ora, può sembrare perfettamente senza significato).

Le parole chiave utilizzabili per definire la sezione marcata sono diverse; di seguito ne appare l’elenco.

Parola chiave	Descrizione
`INCLUDE`	Il contenuto della sezione marcata deve essere incluso nel documento SGML e deve essere elaborato normalmente.
`IGNORE`	Il contenuto della sezione marcata deve essere escluso dal documento SGML. Se l’analizzatore SGML genera un qualche tipo di output, questo non contiene tale sezione.
`CDATA`	Il contenuto della sezione marcata deve essere incluso e trattato come testo letterale, in modo da ignorare ciò che altrimenti potrebbe essere interpretato come un marcatore o un’entità. Ciò vale per tutto, tranne il simbolo di chiusura della sezione marcata (‘**`]]>`**’), che quindi è l’unica cosa che non può essere rappresentata all’interno di questa.
`RCDATA`	Il contenuto della sezione marcata deve essere incluso e trattato come testo letterale, in modo da ignorare ciò che altrimenti potrebbe essere interpretato come un marcatore, ma continuando a espandere le entità.
`TEMP`	Il contenuto della sezione marcata deve essere inteso come «temporaneo». Ciò serve solo come riferimento umano, per localizzare facilmente una parte del documento che richiede una revisione o che deve essere rimossa.

L’utilizzo di sezioni marcate di tipo ‘**`INCLUDE`**’ e ‘**`IGNORE`**’ è utile solo in abbinamento a entità parametriche. Prima di proseguire, è bene chiarire che quella specie di marcatore che apre una sezione marcata è come un’istruzione SGML, di quelle che appaiono nel DTD, anche se viene usata al di fuori di questo, nel documento. In questo senso, al suo interno si possono usare le entità parametriche; quindi, una di queste macro può servire per definire in modo dinamico la parola chiave ‘**`INCLUDE`**’ oppure ‘**`IGNORE`**’, per decidere di includere o escludere quel blocco (e probabilmente anche altri) con la modifica di una sola macro.

Per esempio, nel DTD del documento potrebbe apparire la dichiarazione di un’entità parametrica denominata ‘**`commentato`**’.

```
<!ENTITY % commentato "INCLUDE">
```

Nel documento SGML potrebbero esserci una serie di sezioni marcate la cui inclusione deve dipendere da questa macro.

```
…
1 + 2 = 3
<![%commentato;[
La matematica non è un’opinione.
]]>
…
```

Quando il testo viene analizzato, la macro viene espansa e trovando che corrisponde a ‘**`INCLUDE`**’, il testo delle sezioni marcate che l’hanno usata, vengono incluse. Al contrario, basta modificare la macro, assegnandole il valore ‘**`IGNORE`**’, per fare in modo che tutte quelle sezioni marcate vengano ignorate.

Questo tipo di approccio potrebbe sembrare ugualmente scomodo per l’utilizzatore che non vuole toccare il DTD. Però, come è possibile vedere in seguito, si possono inserire delle eccezioni al DTD nel preambolo di un documento SGML. Oppure, si può benissimo progettare un DTD con una componente esterna, destinata a questo tipo di ritocchi.

51.1.8 Dettagli importanti

Prima di passare alla descrizione dell'abbinamento di un DTD a un testo SGML, è bene chiarire alcuni dettagli che sono stati trascurati fino a questo punto.

51.1.8.1 Commenti

All'interno del documento sorgente SGML, come nel DTD, possono essere indicate delle righe di commento da non considerare parte del documento o della codifica. Queste si ottengono con i delimitatori '**<!--**' e '**-->**'.

Volendo approfondire meglio il problema, la sequenza '**<!>**' rappresenta l'istruzione SGML nulla e può essere usata indifferentemente nel DTD o nel sorgente SGML. In qualità di istruzione nulla viene ignorata semplicemente.

All'interno delle istruzioni SGML è possibile inserire dei commenti, attraverso una sequenza di due trattini ('**--**'), per aprire e chiudere il commento. L'esempio seguente dichiara l'elemento '**itemize**' con un commento incorporato:

```
<!ELEMENT itemize - - (item+) -- elenchi puntati -- >
```

Ciò dovrebbe chiarire il senso del commento composto da '**<!--**' e '**-->**': si tratta di un'istruzione (nulla) che contiene un commento.

Questa particolarità di SGML ha delle conseguente: nel testo che compone il commento, non possono apparire sequenze di due o più trattini.

51.1.8.2 Maiuscole e minuscole

Per convenzione, i nomi di entità sono sensibili alla differenza tra lettere maiuscole e minuscole, per cui '**À**' e '**à**' rappresentano rispettivamente la lettera «A» maiuscola con accento grave e la «a» minuscola con accento grave.

Per convenzione, i nomi degli elementi, i simboli delle regole di minimizzazione, i nomi degli attributi e le parole chiave, non sono sensibili alla differenza tra lettere maiuscole e minuscole. Quindi, nell'ambito delle dichiarazioni del DTD i due esempi seguenti sono identici:

```
<!element ref - o empty>
<!attlist ref
        id cdata #required
        name cdata "riferimento">
```

```
<!ELEMENT REF - O EMPTY>
<!ATTLIST REF
        ID CDATA #REQUIRED
        NAME CDATA "riferimento">
```

Nello stesso modo, nell'ambito del testo SGML sono identici i due esempi seguenti:

```
... <ref id="capitolo-introduttivo" name="Intro"> ...
```

```
... <REF id="capitolo-introduttivo" name="Intro"> ...
```

indifferentemente dal modo (maiuscolo o minuscolo) in cui l'elemento '**ref**' è stato dichiarato nel DTD.

Evidentemente, in generale, il contenuto delle stringhe delimitate è sensibile alla differenza tra maiuscole e minuscole, dal momento che riguarda i programmi che fanno uso del documento dopo l'analisi SGML; in pratica, dipende da questi programmi successivi il senso che hanno tali informazioni.

51.1.8.3 Delimitatori di stringa

In varie situazioni, all'interno del DTD e all'interno dei marcatori utilizzati nel testo SGML, può essere necessaria l'indicazione di stringhe. I simboli utilizzati per delimitare le stringhe possono essere gli apici doppi ('**"**…**"**'), oppure gli apici singoli ('**'**…**'**'). La scelta tra i due tipi di delimitatori dovrebbe essere indifferente, a parte la possibile necessità di inserire nelle stringhe proprio questi caratteri. Si osservi l'esempio seguente, in cui vengono dichiarate le entità riferite ad alcune lettere accentate da usare con LaTeX.


```
<!ENTITY uuml   '\"u'>
<!ENTITY Uuml   '\"U'>
<!ENTITY yacute "\'y">
<!ENTITY Yacute "\'Y">
```

In situazioni più complesse, potrebbe essere necessario indicare i caratteri con l'aiuto delle macro '**&#*nnn*;**', che permettono di identificare l'oggetto attraverso il numero corrispondente riferito al tipo di codifica utilizzato (purché il contesto preveda la successiva ulteriore espansione di tali macro).

51.1.8.4 Tipo di contenuto di un'entità generale

In precedenza, quando è stato mostrato in che modo possa essere definita un'entità, si è trascurato il fatto che si deve definire in che modo la stringa di sostituzione vada interpretata. Per questo, si aggiunge una parola chiave prima della stringa.

Se non si usa alcuna parola chiave, si intende che la stringa vada interpretata come appare, espandendo eventuali entità contenute al suo interno. Si osservi l'esempio.

```
<!ENTITY attenzione "&lt;ATTENZIONE&gt;">
```

Quando dovesse essere utilizzata la macro '**&attenzione;**', si otterrebbe la stringa '**<ATTENZIONE>**', perché le entità '**<**' e '**>**' vengono espanse ulteriormente.

Se si indica la parola chiave '**CDATA**', si intende che la stringa di sostituzione deve essere utilizzata in modo letterale, senza espandere alcuna sequenza che potrebbe sembrare un'entità.

```
<!ENTITY attenzione CDATA "&lt;ATTENZIONE&gt;">
```

L'esempio, modificato con l'introduzione della parola chiave '**CDATA**', fa sì che la macro '**&attenzione;**' si traduca in pratica in '**<ATTENZIONE>**', perché le entità '**<**' e '**>**' non vengono riconosciute come tali e quindi non vengono espanse.

Se si utilizza la parola chiave '**SDATA**' (*Special data*), si intende che la stringa di sostituzione deve essere utilizzata in modo letterale, senza espandere alcuna sequenza che potrebbe sembrare un'entità. Però, a differenza di '**CDATA**', l'informazione viene filtrata in modo particolare quando l'analizzatore SGML genera un risultato transitorio da riutilizzare con un altro programma di composizione.

51.1.8.5 Contenuto elementare degli elementi

In precedenza è già stata spiegata la dichiarazione degli elementi e la dichiarazione del contenuto. In particolare si è visto che attraverso la parola chiave '**#PCDATA**' si fa riferimento a testo normale che viene elaborato normalmente (*parsed*). Ciò significa che questo tipo di testo è soggetto alla sostituzione delle entità, come fino a questo punto si è dato per scontato.

Tuttavia esistono altre parole chiave per definire tipi di testo differenti. Segue l'elenco di quelle più comuni.

Parola chiave	Descrizione
`#PCDATA`	*Parsed character data*. Si riferisce a testo normale soggetto alla sostituzione delle entità. Questo testo non può contenere altri elementi.
`CDATA`	Il contenuto dell'elemento deve essere trattato come testo letterale, in modo da ignorare ciò che altrimenti potrebbe essere interpretato come un marcatore o un'entità. In pratica, la definizione di elementi con contenuto '**CDATA**' è decisamente sconsigliabile. Se esiste la necessità di delimitare una zona di testo da trattare in modo letterale, solitamente, si preferisce utilizzare una sezione marcata del tipo '**<![CDATA[…]]>**'.
`RCDATA`	Il contenuto dell'elemento deve essere trattato come testo letterale, in modo da ignorare ciò che altrimenti potrebbe essere interpretato come un marcatore, ma continuando a espandere le entità.

51.1.9 Abbinare il DTD al documento SGML

L'abbinamento di un DTD a un documento SGML avviene generalmente in modo formale. In presenza di situazioni eccezionali, questo abbinamento può essere implicito, come nel caso dell'HTML, ma è bene utilizzare ugualmente l'approccio generale anche in questi casi estremi.

Un sorgente SGML inizia normalmente con la dichiarazione del tipo di DTD utilizzato. Può trattarsi di un file esterno o di dichiarazioni incorporate nel documento stesso. Per esempio, la dichiarazione seguente indica all'analizzatore SGML di utilizzare un DTD esterno, denominato '**`linuxdoc`**' e contenuto nel file '`linuxdoc.dtd`'.

```
<!DOCTYPE linuxdoc SYSTEM "linuxdoc.dtd">
```

L'esempio seguente mostra invece una dichiarazione iniziale che contiene le istruzioni che compongono il DTD, racchiuse tra parentesi quadre.

```
<!DOCTYPE personale [
...
-- istruzioni SGML --
...
...
]>
```

Una terza possibilità permette di definire un file esterno e di aggiungere altre istruzioni particolari riferite al documento, come nell'esempio seguente, sempre utilizzando le parentesi quadre.

```
<!DOCTYPE linuxdoc SYSTEM "linuxdoc.dtd" [
...
-- istruzioni SGML --
...
...
]>
```

Inoltre, come è stato visto nel caso delle entità, l'acquisizione dall'esterno di un file contenente un DTD, può avvenire anche senza stabilire espressamente il nome di un file, lasciando che questo venga determinato da un catalogo. Così, l'esempio già visto del DTD '**`linuxdoc`**' si potrebbe trasformare nel modo seguente:

```
<!DOCTYPE linuxdoc SYSTEM>
```

Esiste anche un'altra alternativa: quella di indicare un identificatore pubblico, anch'esso riferito a un catalogo. Quello che segue è il preambolo di un file SGML scritto secondo il DTD HTML 3.2. Si osservi, a questo proposito, l'uso della parola chiave '**`PUBLIC`**'.

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

51.1.9.1 Strategie

La scelta di incorporare il DTD nel documento, o di lasciarlo all'esterno, dipende da delle preferenze organizzative. Di sicuro, può essere sensato l'inclusione del DTD nel documento SGML quando si tratta di un DTD specifico che non viene usato altrove.

Nella realtà, si utilizza quasi sempre un DTD esterno, probabilmente predisposto da altri, abbinato a una serie di strumenti che permettono di produrre dei documenti in formato finale a partire dai sorgenti SGML scritti seguendo quel DTD particolare.

L'estensibilità del DTD resta sempre una possibilità utile per poter aggiungere delle entità generali (interne o esterne) o delle entità parametriche allo scopo di gestire opportunamente le sezioni marcate.

51.1.9.2 Entità generali

Come si è visto nella sezione precedente, la dichiarazione del DTD può includere delle istruzioni del DTD, generalmente estendendolo. Questo meccanismo permette, tra le altre cose, di inserire delle entità generali, interne o esterne. Si osservi l'esempio:


```
<!DOCTYPE linuxdoc SYSTEM
[
<!ENTITY pericolo "!!">
<!ENTITY posix    "POSIX">
<!ENTITY unix     "Unix">
<!ENTITY xwin     "X Window System">
<!ENTITY edizione "1999.12.31">
]>
```

L'esempio appena mostrato permette di utilizzare le macro '**`&pericolo;`**', '**`&posix;`**', '**`&unix;`**', '**`&xwin;`**' e '**`&edizione;`**', all'interno del sorgente SGML, ottenendo la loro sostituzione automatica. Per intenderne l'utilità, basta pensare al caso della macro '**`&xwin;`**' dell'esempio precedente: non occorre più ricordare come si deve scrivere («X», «X Window» o «X Window System»); se si decidesse di cambiare, basterebbe modificare la dichiarazione dell'entità. Il concetto è analogo a quello delle macro del precompilatore nei linguaggi di programmazione.

L'esempio fa riferimento in particolare a entità interne, perché tutto quello che serve è contenuto nella dichiarazione, mentre sarebbero entità esterne se si fa riferimento a file esterni per l'espansione del loro contenuto.

Segue un esempio con l'uso di un'entità esterna:

```
<!DOCTYPE linuxdoc SYSTEM
[
<!ENTITY COPY         SYSTEM "formalita/copy.sgml">
]>
```

La definizione di entità generali è consentita anche quando queste dovessero essere già state dichiarate nel DTD. Le entità dichiarate nelle istruzioni aggiuntive, dovrebbero prendere la precedenza e sostituirsi a quelle eventualmente già dichiarate nel DTD.

51.1.9.3 Entità parametriche

Come nel caso delle entità generali, nelle estensioni del DTD può essere conveniente aggiungere la dichiarazione di entità parametriche da utilizzare per controllare l'inclusione o l'esclusione di sezioni marcate. Si osservi l'esempio, già mostrato in precedenza, in cui la macro '**`commentato`**' serve per controllare alcune sezioni marcate, stabilendone il tipo.

```
<!DOCTYPE linuxdoc SYSTEM
[
<!ENTITY pericolo "!!">
<!ENTITY posix    "POSIX">
<!ENTITY unix     "Unix">
<!ENTITY xwin     "X Window System">
<!ENTITY edizione "1999.12.31">
...
<!ENTITY % commentato "INCLUDE">
]>
...
...
1 + 2 = 3
<![%commentato;[
La matematica non è un'opinione.
]]>
...
...
```

51.1.9.4 Ultime annotazioni sulla dichiarazione del DTD

Si è visto in che modo inizia un sorgente SGML. Quello che non è ancora stato chiarito è che il tipo di documento deve essere stato dichiarato nel DTD, anche se ciò può sembrare ridondante. In effetti è necessario dire di cosa è composto il documento. Nel DTD potrebbe apparire un'istruzione come quella seguente:

```
<!ELEMENT linuxdoc O O (article|report|book|letter) >
```

In questo esempio, si comprende che non è necessario usare marcatori del tipo '**`<linuxdoc>`**' '**`</linuxdoc>`**' per delimitare il sorgente SGML. Infatti, la coppia di '**`O`**' afferma che queste sono opzionali.

Invece, il tipo di documento '**linuxdoc**' deve contenere esattamente un elemento del tipo '**article**', oppure '**report**', oppure '**book**', o ancora '**letter**'.

Il sorgente SGML che fa riferimento al tipo di documento '**linuxdoc**' e che utilizza il formato definito dall'elemento '**article**', è composto schematicamente come segue:

```
<!DOCTYPE linuxdoc SYSTEM>
<article>
...
...
...
</article>
```

Un tipo di documento potrebbe essere definito in maniera diversa, per esempio nel modo seguente:

```
<!element miodoc - - ( sezione+ ) >
```

In questo caso, il documento può contenere solo elementi '**sezione**', ed è obbligatorio l'utilizzo dei marcatori per indicare l'inizio e la fine del tipo di documento.

```
<!doctype miodoc system>
<miodoc>
        <sezione>
        ...
        ...
        ...
</miodoc>
```

51.1.10 Mappe di sostituzione (shortref)

Fino a questo punto, si è vista la filosofia dell'SGML applicata alla struttura del documento e all'indipendenza rispetto alla piattaforma. L'analizzatore SGML standard, oltre che convalidare il documento in base al DTD, si occupa di rielaborare il sorgente SGML per generare un risultato intermedio, più facile da gestire per altri programmi di composizione.

In un certo qual modo, questo risultato intermedio può essere controllato, all'interno del DTD, attraverso la definizione di mappe di sostituzione, o *shortref*.

Con questo meccanismo, si punta normalmente ad attribuire significati speciali a simboli determinati, oltre che a controllare la spaziatura orizzontale e verticale del testo.

51.1.10.1 Dichiarazione e abbinamento delle mappe di sostituzione

La mappa di sostituzione definisce un abbinamento tra un simbolo e un'entità che ne deve prendere il posto. L'esempio seguente è solo un pezzo ipotetico della dichiarazione di una mappa del genere.

```
<!SHORTREF miamappa
...
        "[" lsqb
        "]" rsqb
        "~" nbsp
        "_" lowbar
        "#" num
        "%" percnt
        "^" circ
        "{" lcub
        "}" rcub
        "|" verbar >
```

Dall'esempio si può osservare che alcuni simboli vengono sostituiti con le entità relative, indicate solo per nome, senza bisogno della e-commerciale e del punto e virgola finale. Questo fatto, di per sé, potrebbe sembrare assolutamente inutile dal punto di vista di SGML: se si può scrivere una parentesi quadra aperta, perché sostituirla automaticamente con la sua entità corrispondente. Il fatto è che il software utilizzato per la composizione, potrebbe attribuire un significato speciale a una parentesi quadra, mentre quello che si vuole nel testo SGML è che questa valga solo per quello che appare. In tal modo, chi scrive dovrebbe utilizzare necessariamente la macro

'**[**' per non creare problemi al programma di composizione o di elaborazione successiva.

Nello stesso modo, attraverso la mappa di sostituzione, si può attribuire un significato completamente diverso alla parentesi quadra aperta: per assurdo, potrebbe diventare una parentesi graffa...

```
<!SHORTREF miamappa
...
        "[" lcub
        "]" rcub
...
        "{" lcub
        "}" rcub
        "|" verbar >
```

Volendo fare delle acrobazie, si può associare un simbolo a un'entità che poi si traduce in un marcatore. Si osservi l'esempio.

```
<!ENTITY formula1 '<formula>'>
<!ENTITY formula0 '</formula>'>
<!SHORTREF miamappa
...
        "[" formula1
        "]" formula0
...
        "{" lcub
        "}" rcub
        "|" verbar >
```

In questo modo, quando nel testo si utilizzano le parentesi quadre, ciò che si ottiene è l'apertura e la chiusura dell'elemento '**formula**'.

Anche se questa tecnica è stata usata nel noto DTD LinuxDoc, come in Qwertz, proprio per delimitare agevolmente le formule matematiche, si tratta di una cosa decisamente sconsigliabile dal punto di vista dell'SGML.

Gli elementi SGML vanno abbinati alle mappe che si ritiene siano più adatte per i loro scopi. Tuttavia, un elemento può non essere stato abbinato esplicitamente ad alcuna mappa; in tal caso eredita quella dell'elemento che lo contiene effettivamente, di volta in volta, nel documento. Di conseguenza, diventa importante abbinare esplicitamente una mappa almeno all'elemento più esterno, ovvero a quello che corrisponde al nome del tipo stesso di documento.

Dagli esempi mostrati, si può notare che la mappa ha un nome, indicato subito dopo la parola chiave '**SHORTREF**' che apre il comando. Se si vuole abbinare la mappa '**miamappa**' all'elemento '**acronimo**', si procede come nell'esempio seguente:

```
<!USEMAP miamappa acronimo>
```

51.1.10.2 Spaziature e interruzioni di riga

In linea di principio, il risultato dell'elaborazione dell'analizzatore SGML contiene tutti gli spazi orizzontali e verticali esistenti nel sorgente di partenza. Però, per quanto possibile, si cerca normalmente di evitare che il sorgente SGML sia vincolato dalla spaziatura utilizzata, che in realtà potrebbe servire solo per facilitarne la lettura umana con rientri, allineamenti, spazi verticali come si farebbe con un linguaggio di programmazione.

Per questo ci deve essere un modo per poter identificare le spaziature orizzontali, le righe vuote e quelle bianche, in modo da poterle sopprimere nel risultato dell'elaborazione SGML. Naturalmente, bisogna poter distinguere, perché ci sono situazioni in cui gli spazi e le righe vuote hanno un significato e vanno mantenuti.

Per queste cose si utilizzano delle macro speciali, ma prima di descriverle, occorre definire alcuni concetti. Dal punto di vista dell'SGML, una riga è una sequenza di caratteri, con un inizio e una fine, ignorando completamente la codifica che si utilizza in pratica per separare una riga dall'altra.

Nei sistemi Unix, il codice di interruzione di riga è composto dal carattere *<LF>* mentre in altri sistemi si utilizza la sequenza *<CR><LF>*. Per l'SGML è come se questi codici non esistessero:

le righe finiscono **prima** del codice di interruzione di riga e iniziano **dopo** tale codice. Si osservi l'esempio seguente:

```
<paragrafo>Ciao,
come stai?
Io bene; e tu?</paragrafo>
```

L'idea che ha l'SGML di ciò che è stato scritto, può essere rappresentata dallo schema seguente, dove è stato utilizzato il simbolo '**^**' per segnalare l'inizio della riga, il simbolo '**$**' per segnalarne la fine, i simboli '**>**' e '**<**' per indicare l'inizio e la fine dell'elemento.

```
>Ciao,$
^come stai?$
^Io bene; e tu?<
```

Può sembrare strano, ma all'inizio e alla fine del testo mancano questi margini: esiste solo l'inizio e la fine dell'elemento. Se si dovesse sopprimere una riga, si eliminerebbe implicitamente anche il suo inizio e la sua fine.

Da qualche parte si potrebbe leggere che il codice di inizio riga equivale al codice *<LF>*, mentre quello di fine riga corrisponde a *<CR>*. Evidentemente questo ragionamento può valere solo per le piattaforme che utilizzano file di testo con un'interruzione di riga *<CR><LF>*, ma si tratta di una semplificazione che non corrisponde alla logica di SGML e può essere solo forviante.

La tabella 51.70 mostra le macro più importanti che possono essere usate per il controllo delle spaziature superflue.

Tabella 51.70. Simboli di definizione di spaziature e delimitazione delle righe.

Simbolo	Significato
`&#RS;`	Inizio di una riga (*Record start*).
`&#RE;`	Fine di una riga (*Record end*).
`&#RS;B`	Spaziatura iniziale.
`B&#RE;`	Spaziatura finale.
`&#RS;&#RE;`	Una riga vuota.
`&#RS;B&#RE;`	Una riga contenente solo spazi orizzontali (bianca).
`&#SPACE;`	Uno spazio singolo, [*SP*].
`&#TAB;`	Tabulazione, [*HT*].
`BB`	Spazio orizzontale all'interno e agli estremi dell'elemento.

L'esempio seguente mostra una mappa di sostituzione tipica, in cui si vogliono ignorare (e di conseguenza, eliminare) gli spazi orizzontali superflui, le righe vuote, quelle bianche, infine si vuole che tutto il testo si traduca in una riga sola.

```
<!shortref miamappa
        "BB"            space
        "&#RS;B"        null
        "B&#RE;"        space
        "&#RS;B&#RE;"   null
        "&#RS;&#RE;"    null
        "&#RS;"         null
        "&#RE;"         space
        "[" lsqb
        "]" rsqb
        "~" nbsp
        "_" lowbar
        "#" num
        "%" percnt
        "^" circ
        "{" lcub
        "}" rcub
        "|" verbar >
```

Le macro '**&space;**' e '**&null;**' si riferiscono rispettivamente a un solo carattere spazio e alla stringa nulla. Generalmente devono essere dichiarate nel DTD nel modo seguente:


```
<!ENTITY space " ">
<!ENTITY null "">
```

Per comprendere meglio l'effetto della mappa di sostituzione proposta, conviene partire da un esempio e analizzare gli effetti di ogni dichiarazione, una alla volta. In particolare, gli utenti dei sistemi Unix devono dimenticare per un po' il comportamento del codice di interruzione di riga (*new-line*), perché SGML considera solo la stringa nulla all'inizio e alla fine della riga: solo quando la fine di una riga e l'inizio della successiva sono stati rimossi, allora queste due vengono unite assieme.

Supponiamo di cominciare da una variante dell'esempio già descritto, dove sono stati aggiunti tanti spazi orizzontali e verticali superflui.

```
>     Ciao,      $
^$
^         $
^come stai?$
^$
^     Io         bene;         e      tu?     <
```

Applicando la trasformazione '**"BB" space**', vengono sostituiti gli spazi orizzontali all'inizio dell'elemento, alla fine e all'interno delle frasi con uno spazio singolo normale.

```
> Ciao,      $
^$
^         $
^come stai?$
^$
^     Io bene; e tu? <
```

Si può osservare che le frasi si sono ricompattate; inoltre, all'inizio e alla fine dell'elemento è rimasto un solo spazio superfluo (che non può essere rimosso). Si continua applicando '**"&#RS;B" null**'; si ottiene l'eliminazione dell'inizio delle righe (quelle che contengono effettivamente qualcosa) fino al primo carattere diverso da uno spazio orizzontale.

```
> Ciao,      $
^$
^         $
^come stai?$
^$
 Io bene; e tu? <
```

Quando si applica anche '**"B&#RE;" space**'; si ottiene la sostituzione degli spazi orizzontali nella parte finale, fino alla fine delle righe (quelle che contengono effettivamente qualcosa), con uno spazio singolo. Nell'esempio, dal momento che nella prima riga è scomparso il simbolo che segnalava la fine della riga, appare un trattino basso, ma solo per aiutare il lettore.

```
> Ciao,_
^$
^         $
^come stai?$
^$
 Io bene; e tu? <
```

La sostituzione '**"&#RS;B&#RE;" null**' elimina le righe bianche, ma non vuote.

```
> Ciao,_
^$
^come stai?$
^$
 Io bene; e tu? <
```

La sostituzione '**"&#RS;&#RE;" null**' elimina le righe vuote.

```
> Ciao,_
^come stai?$
 Io bene; e tu? <
```

Si può osservare che la riga contenente la frase «come stai?», è rimasta intatta. Infatti, non contenendo spazi aggiuntivi all'inizio o alla

fine, non è mai stata interessata dalle trasformazioni applicate fino a questo momento.

Finalmente entrano in gioco '**"&#RS; null**' e '**"&#RE; space**', per eliminare l'inizio e la fine delle righe rimaste. Per la precisione, la fine delle righe deve essere sostituito con uno spazio singolo, altrimenti si rischia di attaccare assieme delle parole. La trasformazione viene mostrata in due passaggi.

```
> Ciao,_
 come stai?_
 Io bene; e tu? <

> Ciao, come stai? Io bene; e tu? <
```

Nonostante la descrizione fatta con tanta cura, è probabile che la trasformazione '**"&#RS; null**' venga semplicemente ignorata, perché l'analizzatore SGML si limita a tenere in considerazione solo la fine delle righe (*record end*).

51.1.10.3 Limitazioni ed esagerazioni

Da quanto visto nella sezione precedente si potrebbe supporre che il meccanismo delle mappe di sostituzione permetta di sostituire quello che si vuole. Non è così, solo alcuni simboli sono considerati dei possibili *shortref*. In ogni caso, ci si accorge subito quando si usa qualcosa di sbagliato: l'analizzatore SGML avvisa immediatamente.

Attraverso le mappe di sostituzione si possono realizzare anche delle acrobazie che spesso sono poco giustificabili e che sarebbe meglio evitare. A parere di chi scrive, la cosa meno utile che si possa richiedere a un sistema SGML è quella di fare in modo che le righe vuote e quelle bianche nel sorgente siano trasformate in separazioni tra i paragrafi. Infatti, l'SGML non ha questo scopo, eppure molti sistemi si impegnano in questo senso. LinuxDoc raggiunge questo risultato intervenendo proprio nelle mappe di sostituzione, facendo in modo che le righe bianche, identificate dal simbolo '**&#RS;B&#RE;**', e quelle vuote, identificate dal simbolo '**&#RS;&#RE;**', siano sostituite da '** **', ovvero dai marcatori che servono a chiudere e a riaprire un paragrafo.

```
...
<!ENTITY psplit '</p><p>' >
...
<!SHORTREF pmap
        "&#RS;B" null
        "&#RS;B&#RE;" psplit
        "&#RS;&#RE;" psplit
...
        "{" lcub
        "}" rcub
        "|" verbar >
...
```

Quello che si vede sopra è proprio un estratto dal DTD di LinuxDoc, dove si vede che la macro '**&psplit;**' viene poi rimpiazzata dai marcatori già descritti.

Naturalmente, questo non esclude la possibilità di generare una grande quantità di elementi '**p**' vuoti, in presenza di più righe vuote o bianche. È chiaro che, successivamente, il sistema di composizione utilizzato deve prendersi carico della loro eliminazione.

51.1.10.4 Soluzione normale

Dopo aver visto in quanti modi si possono usare le mappe di sostituzione, vale la pena di mostrare una soluzione «normale», in cui il problema che si vuole risolvere è l'eliminazione degli spazi superflui all'inizio e alla fine delle righe, oltre che l'eliminazione delle righe bianche e quelle vuote:


```
<!ENTITY space " ">
<!ENTITY null "">
<!ENTITY recordstart "&#RS;">
<!ENTITY recordend "&#RE;">

<!SHORTREF standard
        "&#RS;B"        recordstart
        "B&#RE;"        recordend
        "&#RS;B&#RE;"   null
        "&#RS;&#RE;"    null
>
```

In questo modo, come si vede, è stato necessario dichiarare due entità nuove, '**recordstart**' e '**recordend**', per poter sopprimere gli spazi iniziali e finali superflui, pur mantenendo la separazione in righe distinte.

51.1.11 Elementi di testo riportato letteralmente

La predisposizione di un elemento SGML che consenta la scrittura di testo da riportare in modo letterale costituisce un problema. Si possono scegliere soluzioni diverse, ma nessuna perfetta secondo tutti i punti di vista.

Questo tipo di problema è particolarmente sentito nella scrittura di documenti tecnici, in cui ci può essere la necessità di mostrare porzioni di codice scritto in un qualche linguaggio di programmazione. Per evitare che simboli determinati vengano interpretati dall'analizzatore SGML, occorrerebbe utilizzare continuamente delle macro alternative.

Si possono seguire due direzioni per cercare di risolvere il problema: l'uso di elementi predisposti per un tipo di contenuto più o meno letterale, oppure l'uso di elementi normali con l'aggiunta di una sezione marcata di tipo '**CDATA**'.

51.1.11.1 Tipo di contenuto letterale

Nella definizione di un elemento occorre stabilite il tipo di contenuto. A livello elementare, quando l'elemento non può contenere altri elementi, si utilizza normalmente la parola chiave '**#PCDATA**', con cui si fa riferimento a testo che viene analizzato alla ricerca di entità generali da espandere, senza ammette altri elementi al suo interno.

Per ottenere un elemento adatto al contenuto letterale, si usa solitamente il tipo di contenuto definito dalla parola chiave '**RCDATA**', che non è perfettamente letterale, ma vi si avvicina molto. Per la precisione, la forma del testo viene mantenuta, con tutte le sue spaziature e le interruzioni di riga, ma le entità generali vengono espanse, mentre vengono ignorati eventuali marcatori di apertura. Ciò significa che, la e-commerciale ('**&**') non può essere usata in modo letterale, a meno di usare una macro adatta al suo posto. Lo stesso ragionamento riguarda la sequenza di minore e barra obliqua ('**</**'), che è ammessa solo nel marcatore di chiusura di questo elemento.

```
<!ELEMENT formattato - - RCDATA>
```

L'esempio mostra la dichiarazione dell'elemento '**formattato**', di tipo '**RCDATA**'. Generalmente, per poter utilizzare questo elemento nel modo corretto, si devono dichiarare anche due entità generali specifiche.

```
<!ENTITY ero   CDATA "&">
<!ENTITY etago '</' >
```

In tal modo, al posto del simbolo '**&**' si deve utilizzare la macro '**&ero;**', mentre al posto della sequenza '**</**', si deve usare la macro '**&etago;**'. È il caso di osservare che l'entità generale '**ero**' è volutamente diversa da un'entità analoga, necessaria a indicare una e-commerciale in un testo normale. Infatti, in questo caso, si vuole generare un testo letterale, che si presume possa essere interpretato nello stesso modo letterale anche da altro software di composizione successivo.

In alternativa, si potrebbe usare anche un tipo di contenuto definito dalla parola chiave '**CDATA**', che dovrebbe essere in grado di ignorare sia i simboli dei marcatori, che le macro delle entità generali.

Di fatto però, questo tipo di elemento non dà normalmente i risultati sperati.

51.1.11.2 Sezioni marcate

Nel sorgente SGML, all'interno di un elemento che non sia stato predisposto per un contenuto letterale, è possibile inserire una sezione marcata di tipo '**CDATA**', come nell'esempio seguente:

```
<![CDATA[
Testo letterale: &amp;, &etago;, <ciao>, </ciao>,
ecc., vengono trattati in modo letterale.
]]>
```

In tal modo, vengono preservati anche gli spazi, orizzontali e verticali, e ogni eventuale mappa di sostituzione (*shortref*) viene ignorata temporaneamente. L'unica cosa che non può contenere questo ambiente, è la sequenza '**]]>**', che serve a concludere la sezione marcata.

Bisogna tenere presente che la sequenza '**]]>**' può essere rappresentata anche con l'inserzione di spazi; per esempio come '**]]>**', '**]] >**' o '**]] >**', che rappresentano sempre la stessa cosa.

Questa tecnica ha il vantaggio di potersi applicare anche a un DTD che non sia stato predisposto con elementi atti all'inserimento di testo letterale. Purtroppo, non tutti gli strumenti SGML sono in grado di riconoscere le sezioni marcate; si pensi ai navigatori, che pur sapendo interpretare l'HTML, non sono sempre in grado di riconoscere tali particolarità.

51.1.12 Cataloghi

Nelle sezioni precedenti si è visto che il DTD può essere composto da diversi file fisici nel sistema. Lo stesso preambolo di un sorgente SGML prevede la dichiarazione e l'inclusione di un DTD. È stato mostrato come includere un blocco di DTD esterno, attraverso la dichiarazione e il successivo utilizzo di un'entità parametrica che fa riferimento a un file esterno.

Quando si vogliono utilizzare componenti esterni senza fare riferimento a un file preciso, si possono predisporre dei cataloghi, con i quali si esplicitano questi dettagli riferiti al sistema di cui si dispone effettivamente.

Questo tipo di approccio viene usato tipicamente per due motivi: evitare di dover fare riferimento a un file preciso per il DTD nella dichiarazione del tipo di documento all'inizio del sorgente SGML; includere in modo dinamico le entità standard riferite alle lettere accentate e ai simboli speciali. Per quanto riguarda il secondo problema, si deve tenere presente che l'SGML si astrae dalla piattaforma, quindi, il modo in cui le entità di questo tipo vanno rappresentate dipende da quello che si vuole fare dopo.

51.1.12.1 Riferimenti esterni

Generalmente, quando si vogliono usare i cataloghi, si possono fare due tipi di riferimenti a componenti esterne: l'identificatore pubblico e l'identificatore di sistema. Seguono quattro esempi significativi a questo proposito: nei primi due si tratta della dichiarazione del tipo di documento '**HTML**' e di un'entità parametrica, attraverso un identificatore pubblico (una stringa piuttosto lunga); negli ultimi due si tratta delle stesse dichiarazioni, ma fatte attraverso un identificatore di sistema.

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

```
<!ENTITY % ISOlat1 PUBLIC "ISO 8879:1986//ENTITIES Added Latin 1//EN">
```

```
<!DOCTYPE HTML SYSTEM>
```

```
<!ENTITY % ISOlat1 SYSTEM>
```

Si tratta, evidentemente, di due approcci equivalenti, ma che hanno delle conseguenze differenti nell'applicazione pratica. Dalle parole

chiave utilizzate, '**PUBLIC**' e '**SYSTEM**', si può intuire che l'identificatore di sistema è legato alla situazione del sistema, anche se non è obbligatoria l'indicazione immediata del file corrispondente.

L'uso degli identificatori pubblici è quindi una scelta più conveniente, essendo meno vincolata alla piattaforma. Infatti, questi vengono utilizzati prevalentemente per tutto ciò che è già stato standardizzato: i DTD standard e le entità esterne standard.

51.1.12.2 Il catalogo in pratica

Quando si usano strumenti di analisi ed elaborazione SGML comuni, il catalogo è un file. A seconda degli strumenti utilizzati, potrebbe essere necessario configurare una variabile di ambiente, o usare un'opzione opportuna nella riga di comando, per comunicare a questi la sua posizione.

Il catalogo serve a esplicitare tutte le componenti esterne che non sono state indicate in modo preciso (il nome del file). Si osservi l'esempio seguente:

```
-- Entità standard richiamate attraverso un identificatore di sistema --
-- Sarebbe meglio non usare questo metodo --
ENTITY %ISOlat1            "ISOlat1"
ENTITY %ISOnum             "ISOnum"
ENTITY %ISOdia             "ISOdia"

-- Entità standard richiamate attraverso un identificatore pubblico --
-- Questo tipo di indicazione è preferibile in generale --
PUBLIC "ISO 8879:1986//ENTITIES Added Latin 1//EN"                "ISOlat1"
PUBLIC "ISO 8879:1986//ENTITIES Numeric and Special Graphic//EN"  "ISOnum"
PUBLIC "ISO 8879:1986//ENTITIES Diacritical Marks//EN"            "ISOdia"

-- DTD predefinito per il tipo HTML --
DOCTYPE "HTML"                                  "html32.dtd"

-- Identificatori pubblici per le varie forme dell'HTML 3.2 --
PUBLIC "-//W3C//DTD HTML 3.2//EN"               "html32.dtd"
PUBLIC "-//W3C//DTD HTML 3.2 Draft//EN"         "html32.dtd"
PUBLIC "-//W3C//DTD HTML 3.2 Final//EN"         "html32.dtd"
```

Ogni direttiva dell'esempio occupa una riga e si compone di tre parti, dove l'ultima informazione rappresenta il file da utilizzare per quel particolare tipo di entità, documento o identificatore pubblico.

Per la precisione, invece che di file, occorrerebbe parlare di identificatore di sistema effettivo, dove questo concetto viene poi definito dallo standard ISO 8879. In generale si tratta di file e questo dovrebbe bastare come primo approccio all'SGML.

Si noti che i commenti sono delimitati da coppie di trattini, '**--**', come si fa all'interno delle istruzioni SGML.

Direttiva	Descrizione
`PUBLIC` ***identificatore_pubblico*** ↩ ↪***identificatore_di_sistema***	Stabilisce l'identificatore di sistema effettivo (il file) corrispondente all'identificatore pubblico indicato. Quando possibile, è preferibile utilizzare gli identificatori pubblici per definire gli oggetti.
`DOCTYPE` ***nome identificatore_di_sistema***	Stabilisce l'identificatore di sistema effettivo (il file) corrispondente al nome del tipo di documento indicato. Dal momento che questo nome può fare riferimento a uno tra diversi DTD alternativi (si pensi al caso dell'HTML con le sue versioni), questa dichiarazione serve prevalentemente per stabilire un DTD predefinito nel caso in cui non sia stato specificato un identificatore pubblico nel documento che si elabora.

Direttiva	Descrizione
`ENTITY` ***nome identificatore_di_sistema***	Stabilisce l'identificatore di sistema effettivo (il file) corrispondente all'entità generale indicata.
`ENTITY %`***nome identificatore_di_sistema***	Stabilisce l'identificatore di sistema effettivo (il file) corrispondente all'entità parametrica indicata. Si osservi il fatto che il simbolo di percentuale è attaccato al nome dell'entità.

Negli esempi seguenti, viene mostrata prima l'istruzione utilizzata nel DTD, o nel preambolo del sorgente SGML, quindi si presenta la direttiva corrispondente, necessaria nel catalogo.

- `<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">`
  Si tratta della dichiarazione, all'inizio di un sorgente SGML, dell'utilizzo del DTD '**`HTML`**', definito in base all'identificatore pubblico '**`-//W3C//DTD HTML 3.2 Final//EN`**'. Perché da questo si possa arrivare a identificare un file particolare, occorre che nel catalogo appaia una direttiva come quella seguente:
  ```
  PUBLIC "-//W3C//DTD HTML 3.2 Final//EN"          "html32.dtd"
  ```
  In tal caso, all'identificatore pubblico '**`-//W3C//DTD HTML 3.2 Fin`** viene abbinato il file '`html32.dtd`'.
- `<!DOCTYPE HTML SYSTEM">`
  Si tratta della dichiarazione, all'inizio di un sorgente SGML, dell'utilizzo del DTD '**`HTML`**', utilizzando un identificatore di sistema che non viene precisato in modo effettivo. Perché da questo si possa arrivare a identificare un file particolare, occorre che nel catalogo appaia una direttiva come quella seguente:
  ```
  DOCTYPE HTML            "html32.dtd"
  ```
  In tal caso, all'identificatore di sistema '**`HTML`**', viene abbinato il file '`html32.dtd`' (l'identificatore di sistema effettivo).
- `<!ENTITY % ISOlat1 PUBLIC "ISO 8879:1986//ENTITIES Added Latin 1//EN">`
  All'interno del DTD, dichiara l'entità parametrica '**`ISOlat1`**' definita secondo l'identificatore pubblico '**`ISO 8879:1986//ENTITIES Added Latin 1//EN`**'. Perché da questo si possa arrivare a identificare un file particolare, occorre che nel catalogo appaia una direttiva simile a una delle due mostrate di seguito:
  ```
  PUBLIC "ISO 8879:1986//ENTITIES Added Latin 1//EN" "ISOlat1.latex"
  ```
  ```
  ENTITY %ISOlat1 "ISOlat1.latex"
  ```
  Nel primo caso, all'identificatore pubblico '**`ISO 8879:1986//ENTITIES Added Latin 1//EN`**', viene abbinato il file '`ISOlat1.latex`'; nel secondo, si specifica direttamente che l'entità parametrica '**`ISOlat1`**' corrisponde al contenuto del file '`ISOlat1.latex`'.
- `<!ENTITY % ISOlat1 SYSTEM>`
  Nel DTD, viene dichiarata l'entità parametrica '**`ISOlat1`**', utilizzando un identificatore di sistema che non viene precisato in modo effettivo. Perché da questo si possa arrivare a identificare un file particolare, occorre necessariamente che nel catalogo appaia una direttiva come quella seguente, dal momento che non è possibile fare riferimento a un identificatore pubblico:
  ```
  ENTITY %ISOlat1              "ISOlat1.latex"
  ```
  In questo modo si definisce l'identificatore di sistema effettivo dell'entità parametrica '**`ISOlat1`**', facendola corrispondere al file '`ISOlat1.latex`' (l'identificatore di sistema effettivo).

51.2 Elaborazione SGML

L'elaborazione SGML si compone fondamentalmente di un programma in grado di verificare la correttezza formale di un sorgente SGML in base al suo DTD. Questo tipo di programma è l'analizzatore SGML (*SGML parser*) e il suo compito si estende frequentemente alla generazione di un risultato intermedio, pronto per una rielaborazione successiva, normalmente attraverso un sistema di composizione tipografica.

L'elaborazione successiva richiede strumenti specifici, ma per le situazioni più semplici, dove basta rimpiazzare un marcatore con una codifica equivalente adatta a un programma di composizione tipografica particolare, si è utilizzato in passato il cosiddetto ASP: *Amsterdam SGML parser*.

L'utilizzo di un analizzatore SGML, precisamente il pacchetto SP con il programma '**`nsgmls`**', è una cosa consueta e attuale, mentre l'utilizzo di un analizzatore ASP può considerarsi una tecnica superata. Tuttavia, l'abbinamento di '**`nsgmls`**' e '**`sgmlsasp`**' (il secondo è un analizzatore ASP) è un metodo semplice e pratico per costruire i propri strumenti SGML, quando non si vuole utilizzare quello che è già a disposizione.

In sostituzione di '**`sgmlsasp`**' si può utilizzare anche il pacchetto SGMLSpm, il quale si compone di una serie di moduli Perl e in particolare fornisce il programma '**`sgmlspl`**', che svolge un compito simile a quello di un analizzatore ASP.

51.2.1 SP

SP è il pacchetto di analisi SGML di James Clark. Si tratta dello strumento fondamentale, ed è disponibile anche su piattaforme differenti dai sistemi Unix. In passato, al posto di SP, era disponibile il pacchetto Sgmls che comunque non è compatibile con molte caratteristiche particolari dell'SGML.

Il pacchetto SP contiene il programma '**`nsgmls`**', assieme a una serie di DTD di esempio. Il programma '**`nsgmls`**' è tutto quello che serve per convalidare un file SGML con il suo DTD e per generare un risultato intermedio analizzabile automaticamente attraverso '**`sgmlsasp`**', un accessorio del vecchio pacchetto Sgmls, o in alternativa attraverso '**`sgmlspl`**', del pacchetto SGMLSpm.

51.2.1.1 Avvio di nsgmls

Il programma '**`nsgmls`**' utilizza lo standard input, oppure i file indicati in coda alla riga di comando (gli identificatori di sistema), per analizzarne il contenuto secondo l'SGML ed eventualmente per generare un output pre-elaborato.

```
nsgmls [opzioni] [identificatore_di_sistema]...
```

Gli errori vengono segnalati attraverso lo standard error, mentre il risultato dell'elaborazione viene emesso attraverso lo standard output.

Opzione	Descrizione
`-c` ***identificatore_di_sistema***	Permette di specificare l'utilizzo di un catalogo, rappresentato dal file indicato come argomento dell'opzione. Questa opzione può essere specificata più volte, per richiedere l'utilizzo di più cataloghi. Se nella stessa directory del file del documento analizzato esiste un file denominato '`catalog`', questo viene aggiunto in coda ai cataloghi letti attraverso questa opzione. Inoltre, se esiste la variabile di ambiente ***SGML_CATALOG_FILES***, l'elenco dei cataloghi in essa contenuti viene aggiunto in coda a tutti gli altri.

Opzione	Descrizione
`-D` ***directory***	Permette di definire una directory da utilizzare per la ricerca di file specificati negli identificatori di sistema. Sono ammissibili più opzioni '**`-D`**'. Se esiste la variabile di ambiente ***SGML_SEARCH_PATH***, l'elenco di directory che questa contiene viene aggiunto in coda a quello definito attraverso l'opzione '**`-D`**'.
`-E` ***n_massimo_errori***	Permette di stabilire il numero massimo di errori, dopo il quale '**`nsgmls`**' termina l'analisi. Il valore predefinito è 200.
`-i`***nome***	Permette di definire un'entità parametrica, con il nome indicato, contenente la stringa '**`INCLUDE`**'. In pratica ciò che nel DTD dovrebbe essere definito con l'istruzione '**`<!ENTITY %` *nome* `"INCLUDE">`**'. Questa dichiarazione prende la precedenza su un'altra dichiarazione della stessa entità fatta in qualunque altra posizione; il suo scopo è quello di facilitare la gestione delle sezioni marcate da includere in modo condizionato. In pratica, si definiscono nel DTD solo entità parametriche di questo tipo con il valore '**`IGNORE`**', con le quali si delimitano parti di testo attraverso l'uso di sezioni marcate. Quindi, quando si vogliono includere quelle porzioni di testo, si può utilizzare questa opzione, anche più volte, per fare sì che le entità parametriche desiderate contengano invece la parola chiave '**`INCLUDE`**'.
`-s`	Sopprime l'emissione dell'output intermedio. In questo modo si limita a emettere le segnalazioni di errori attraverso lo standard error.
`-p`	Analizza solo il prologo, in pratica il DTD, ignorando il documento. Ciò implica, di fatto, l'uso dell'opzione '**`-s`**'.

Segue la descrizione di alcuni esempi.

- `$` **`nsgmls -s -c ~/catalogo`** [*Invio*]

  Si limita a convalidare il contenuto del documento proveniente dallo standard input, avvalendosi del catalogo contenuto del file '`~/catalogo`'.

- `$` **`nsgmls -c ~/catalogo`** [*Invio*]

  Convalida il contenuto del documento proveniente dallo standard input, avvalendosi del catalogo contenuto del file '`~/catalogo`', generando anche il documento rielaborato opportunamente.

- `$` **`nsgmls -i annotazioni -c ~/catalogo`** [*Invio*]

  Come nell'esempio precedente, ma in più dichiara l'entità parametrica '**`annotazioni`**' contenente la parola chiave '**`INCLUDE`**'.

51.2.1.2 Variabili di ambiente

Ci sono due variabili di ambiente a cui è sensibile '**`nsgmls`**': ***SGML_SEARCH_PATH*** e ***SGML_CATALOG_FILES***. Entrambe servono a contenere l'indicazione di un elenco di percorsi, separati attraverso i soliti due punti ('**`:`**').

La variabile ***SGML_SEARCH_PATH*** serve ad aggiungere altre directory a quelle che possono essere definite attraverso l'opzione '**`-D`**', per la ricerca di file corrispondenti agli identificatori di sistema.

La variabile ***SGML_CATALOG_FILES*** serve ad aggiungere altri cataloghi (indicati con il loro percorso assoluto) a quelli che possono essere definiti attraverso l'opzione '**`-c`**'.

Queste due variabili possono essere molto importanti quando si devono fornire queste indicazioni, senza avere il controllo diretto sul comando di avvio dell'eseguibile '**`nsgmls`**'. In pratica, quando si installano strumenti SGML che si avvalgono di SP e c'è la necessità di indicare dove si trova il file del catalogo, oppure dove si trovano gli altri file, la modifica di queste variabili può essere l'unica soluzione.

51.2.1.3 Formato dell'output

Il risultato dell'output dell'elaborazione di un file SGML attraverso '**`nsgmls`**' è composto da una serie di righe di testo, di lunghezza variabile, precedute da un carattere nella prima colonna che ne definisce il significato.

In pratica, ogni riga inizia necessariamente con un codice composto da un solo carattere di «comando», e subito dopo, senza spazi aggiuntivi, inizia il contenuto di uno o più argomenti, a seconda del comando, separati da un solo carattere spazio. L'ultimo argomento (che potrebbe anche essere l'unico) può contenere spazi.

Gli «argomenti» di questi comandi possono contenere delle sequenze di escape:

Sequenza	Descrizione
`\\`	rappresenta una singola barra obliqua inversa ('\');
`\n`	rappresenta la fine di una riga (*record end*), secondo la logica di SGML;
`\\|`	viene usato per delimitare (all'inizio e alla fine) le entità di tipo '**`SDATA`**', dopo che queste sono state espanse regolarmente;
`\`***nnn***	rappresenta un carattere particolare attraverso il suo codice ottale.

'**`nsgmls`**' prevede un numero molto grande di caratteri di comando per distinguere il contenuto delle righe del risultato dell'elaborazione. Qui ne vengono mostrati solo alcuni, i più comuni. Gli altri sono descritti dettagliatamente nella pagina di manuale *nsgmls(1)*.

Comando	Descrizione
`(`***identificatore_generico***	Una parentesi aperta rappresenta l'inizio di un elemento, nominato subito dopo (l'identificatore generico). Se questo elemento dovesse avere attributi, verrebbero rappresentati prima, attraverso i comandi '**`A`**'.
`)`***identificatore_generico***	Una parentesi chiusa rappresenta la fine di un elemento, nominato subito dopo (l'identificatore generico).
`A`***nome_attributo valore***	Specifica un attributo per il prossimo elemento. Se l'elemento possiede più attributi, si utilizzano altrettanti record di tipo '**`A`**'. Il valore assegnato all'attributo si può articolare in più componenti, che qui non vengono descritte.
`C`	Questa lettera, che appare da sola alla fine dell'output di '**`nsgmls`**', rappresenta che il contenuto del file sorgente è corretto.

Di seguito vengono descritti alcuni esempi, rappresentati da pezzi dell'output di '**`nsgmls`**'.

- ```
  (HTML
  (HEAD
  (TITLE
  -Introduzione all'SGML
  )TITLE
  )HEAD
  (BODY
  ...
  )BODY
  )HTML
  ```

  Quello che si vede sopra, rappresenta lo schema fondamentale di ciò che si può ottenere analizzando un file HTML. Si può osservare l'apertura e la chiusura dei vari elementi ('**`HTML`**', '**`HEAD`**', '**`TITLE`**', '**`HEAD`**', '**`BODY`**'). I puntini di sospensione rappresentano solo l'interruzione e la ripresa della visualizzazione dell'output.
  ```

```
(P
-Ciao,\ncome stai?\nIo bene; e tu?
)P
```

Rappresenta un elemento '**P**' contenente una frase, divisa in vari punti dal codice '**\n**', che rappresenta la fine della riga (*record end*) secondo SGML.

```
ANAME IMPLIED
AHREF CDATA indice.html
AREL IMPLIED
AREV IMPLIED
ATITLE IMPLIED
(A
-Indice generale
)A
```

Rappresenta un elemento '**A**', contenente la frase «Indice generale», e una serie di attributi: '**NAME**', '**HREF**', '**REL**', '**REV**' e '**TITLE**'.

```
C
```

Alla fine dell'output, il carattere di comando '**C**' rappresenta il buon fine dell'elaborazione.

51.2.2 Sgmls

Il pacchetto Sgmls è stato il predecessore di SP. Questo forniva il programma '**sgmls**', il cui funzionamento è analogo a '**nsgmls**' anche se meno completo, e '**sgmlsasp**', un analizzatore ASP utile ancora adesso in quanto abbinabile all'output di '**nsgmls**'.

'**sgmlsasp**' elabora lo standard input, in base al contenuto di uno o più file specificati come argomenti. Lo standard input deve essere compatibile con il formato standard di '**sgmls**' e di '**nsgmls**', mentre i file di rimpiazzo devono rispettare il formato ASP (*Amsterdam SGML parser*). Il risultato viene emesso attraverso lo standard output.

```
sgmlsasp file_di_rimpiazzo...
```

'**sgmlsasp**' è in grado di elaborare solo alcuni dei comandi contenuti nei record dell'output di '**nsgmls**', cosa che limita in parte le funzionalità utilizzabili con l'SGML.

Il file di rimpiazzo, secondo lo standard ASP, permette di sostituire i marcatori riferiti alle entità con delle stringhe che si presume siano utili per l'elaborazione successiva del testo. Questo file può contenere dei commenti, preceduti dal simbolo di percentuale e terminati dalla fine della riga del file. Le righe bianche e quelle vuote vengono ignorate.

Le direttive si compongono di due soli elementi: il marcatore di apertura o di chiusura e la stringa da utilizzare per il rimpiazzo. Si osservi l'esempio seguente:

```
<titolo>        +       "\n\\section{"
</titolo>               "}"                     +
```

In questo modo, si dichiara di voler sostituire il marcatore '**<titolo>**' con la stringa '**\n\\section{**', mentre il marcatore '**</titolo>**' va sostituito con la stringa '**}**'. Come può intuire chi conosce LaTeX, si vuole sostituire all'elemento '**titolo**' l'ambiente '**\section{}**' di LaTeX.

La stringa usata per il rimpiazzo può contenere delle sequenze di escape. Per la precisione può trattarsi di:

Sequenza	Descrizione
`\\`	rappresenta una singola barra obliqua inversa ('\');
`\"`	rappresenta un apice doppio con valore letterale;
`\[`	rappresenta una parentesi quadra aperta con valore letterale;
`\]`	rappresenta una parentesi quadra chiusa con valore letterale;
`\n`	rappresenta un'interruzione di riga (*newline*).

Pertanto, la stringa di rimpiazzo vista nell'esempio, va letta come: '*new-line***\section{**'.

All'inizio e alla fine della stringa di rimpiazzo può apparire il segno '**+**'. Se è presente, significa che in quel punto si richiede espressamente l'aggiunta di un'interruzione di riga. Se una stringa di rimpiazzo termina con un '**+**' e subito dopo si deve inserire un'altra stringa di rimpiazzo che è preceduta da un altro '**+**', si ottiene comunque una sola interruzione di riga, perché il secondo '**+**' si limita a confermarla.

Una stringa di rimpiazzo può apparire su più righe, come nell'esempio seguente:

```
<relazione>     +       "\\documentstyle{article}\n"
                        "\\begin{document}"     +

</relazione>    +       "\\end{document}"       +
```

Quando un elemento prevede degli attributi, il contenuto di questi può essere inserito nella stringa di rimpiazzo utilizzando la notazione '**[***nome_attributo***]**', dove le parentesi quadre servono a delimitare questo nome, che in particolare va indicato con caratteri maiuscoli.

```
<etichetta>             "\\label{[ID]}"
</etichetta>
```

L'esempio mostra la sostituzione del marcatore '**<etichetta id=...>**' con la stringa '**\label{...}**', dove i puntini di sospensione rappresentano il valore dell'attributo '**ID**'.

51.2.3 SGMLSpm

SGMLSpm è un pacchetto che si compone di moduli e programmi Perl, per la gestione dell'output generato da '**nsgmls**' (SP). Il modo più semplice per sfruttare le funzionalità di questo pacchetto è quello di utilizzare direttamente il programma '**sgmlspl**', scritto ovviamente in Perl, con cui è sufficiente predisporre un file simile a quello utilizzato per la sostituzione ASP.

Qui viene mostrato soltanto il funzionamento di '**sgmlspl**', ma il lettore tenga presente che il pacchetto SGMLSpm offre molte possibilità in più, se si vuole programmare in Perl allo scopo di elaborare l'SGML.

```
sgmlspl script [opzione_script]... < file_sp > file_elaborato
```

'**sgmlspl**' elabora quanto riceve dallo standard input generando un risultato che emette attraverso lo standard output, utilizzando le specifiche indicate nel file che deve essere indicato come primo e unico argomento, che in pratica è uno script di '**sgmlspl**' stesso.

In questo senso, eventuali argomenti successivi vengono passati direttamente allo script.

In pratica, lo standard input deve corrispondere al risultato emesso dall'analizzatore SP ('**nsgmls**') e il file delle specifiche è un pezzo di programma Perl, scritto sfruttando le caratteristiche di SGMLSpm. È il file delle specifiche che stabilisce il modo in cui i marcatori degli elementi SGML vengono trasformati nel risultato finale.

51.2.3.1 File con le specifiche di sostituzione

Rispetto al meccanismo di rimpiazzo utilizzato da ASP, in questo caso si devono scrivere delle righe di codice Perl abbinate agli eventi che interessano, riferiti all'analisi del file generato da SP. Volendo, oltre a distinguere i marcatori di apertura e di chiusura degli elementi, si possono individuare anche le stringhe SDATA e altri componenti di utilizzo meno frequente. Tenendo conto che il pacchetto SGMLSpm è accompagnato da una buona documentazione, qui viene mostrato semplicemente come gestire la sostituzione dei marcatori che delimitano gli elementi SGML.

Come accennato, il file per la sostituzione (ovvero il file delle specifiche) è scritto in Perl e, in particolare, tutto è visto in forma di reazione al verificarsi di un evento:

```
sgml( evento, funzione_da_eseguire );
```

Quello appena mostrato è lo schema generale delle istruzioni da utilizzare per descrivere ciò che deve fare '**sgmlspl**' quando si verifica l'evento specificato nel primo argomento. In pratica, quando si verifica, viene eseguita la funzione del secondo argomento.

L'evento viene specificato in forma di stringa, dove in particolare la forma '**<*x*>**' rappresenta l'incontro del marcatore di apertura dell'elemento ***x*** e, conseguentemente, '**</*x*>**' rappresenta il marcatore di chiusura. Naturalmente, '**sgmlspl**' è in grado di intercettare molti altri tipi di eventi, che comunque non vengono mostrati qui.

È importante tenere presente che gli eventi che identificano i marcatori di apertura e di chiusura degli elementi SGML, devono essere indicati nella loro forma «normalizzata» secondo l'SGML. In pratica, ciò significa che in generale devono essere annotati utilizzando esclusivamente lettere maiuscole.

La funzione indicata come secondo argomento può essere semplicemente una stringa, intendendo che questa rappresenti ciò che si vuole emettere al posto dell'evento che si è manifestato, oppure una funzione (eventualmente un puntatore a una funzione dichiarata altrove), che probabilmente si occupa di generare un qualche tipo di output.

Generalmente, all'interno delle funzioni da abbinare agli eventi si utilizza la subroutine '**output**' per emettere dell'output, secondo quanto prescritto dalla documentazione di SGMLSpm.

Il passaggio degli attributi contenuti eventualmente nei marcatori di apertura degli elementi SGML, non è così intuitivo come avviene nella sintassi ASP. In questo caso occorre considerare che la funzione indicata come secondo argomento riceve degli argomenti in forma di oggetti, da cui possono essere estratte le informazioni sugli attributi SGML.

Si passa alla dimostrazione di alcuni esempi che dovrebbero essere sufficienti per mostrare l'utilizzo essenziale del file delle specifiche di sostituzione per '**sgmlspl**'.

- ```
  sgml( '<RELAZIONE>', "\n\\documentstyle{article}\n\\begin{document}\n" );
  sgml( '</RELAZIONE>', "\n\\end{document}\n");
  ```

  Questa è la situazione più semplice, in cui ci si limita a sostituire i marcatori con una stringa conveniente (in questo caso si tratta di istruzioni LaTeX). Si osservi il fatto che le istruzioni terminano con il punto e virgola; inoltre si utilizza la sequenza '**\n**' per indicare l'inserimento di un codice di interruzione di riga.

- ```
  sgml( '<RELAZIONE>', sub {
      output "\n\\documentstyle{article}";
      output "\n\\begin{document}\n";
  });
  sgml( '</RELAZIONE>', sub {
      output "\n\\end{document}\n";
  });
  ```

  In questo caso, si vuole ottenere lo stesso risultato dell'esempio precedente, con la differenza che nel secondo argomento si indica effettivamente una funzione (senza nome), il cui scopo è semplicemente quello di emettere le stesse stringhe già viste precedentemente, attraverso la subroutine '**output**'.

- ```
  sub relazione_apertura {
      output "\n\\documentstyle{article}";
      output "\n\\begin{document}\n";
  });

  sgml( '<RELAZIONE>', \&relazione_apertura );
  ```

  Questa rappresenta un'altra variante dell'esempio iniziale, in cui, per il marcatore di apertura, si fa riferimento a una subroutine esterna, indicata attraverso un puntatore alla stessa.

- ```
  sgml( '<ETICHETTA>', sub{
      my ($elemento,$evento) = @_;
      my $id = $elemento->attribute('ID')->value;
      output "\\label{$id}";
  });
  sgml( '</ETICHETTA>', '' );
  ```

  Questo esempio mostra il caso di un elemento SGML che prevede l'attributo '**ID**' nel marcatore di apertura. Per estrarre il valore di questo attributo occorre agire come si vede: si distinguono gli argomenti della funzione dichiarando due variabili private corrispondenti, '**my ($elemento,$evento) = @_;**', quindi si ottiene l'attributo richiesto dall'oggetto a cui fa riferimento la variabile '**$elemento**': '**$elemento->attribute('ID')->value**'. Quello che si ottiene viene conservato nella variabile '**$id**', che poi viene inserita nella stringa emessa attraverso la subroutine '**output**'.

  In questo caso, il marcatore di chiusura dell'elemento viene rimpiazzato semplicemente con una stringa nulla.

- ```
  sgml( '<IMMAGINE>', sub{
      my ($elemento,$evento) = @_;
      my $file = $elemento->attribute('FILE')->value;
      my $altezza = $elemento->attribute('ALTEZZA')->value;
      output "\n\\begin{center}\n";
      output "\\epsfig{file=$file,height=$altezza,angle=0}\n";
      output "\\end{center}\n";
  });
  sgml( '</IMMAGINE>', '' );
  ```

  Quello che si vede è un esempio simile a quello precedente, con la differenza che gli attributi da estrarre sono due.

- ```
  sgml('<LIST>', sub {
    my ($element,$event) = @_;
    my $type = $element->attribute('TYPE')->value;

    if ($type eq 'ORDERED') {
      output "\\begin{enumerate}\n";
    } elsif ($type eq 'UNORDERED') {
      output "\\begin{itemize}\n";
    } else {
      die "Bad TYPE '$type' for element LIST at line " .
        $event->line . " in " . $event->file . "\n";
    }
  });
  ```

  Questo esempio proviene dalla documentazione di SGMLSpm e mostra in che modo modificare il risultato della trasformazione in base al contenuto degli attributi di un elemento SGML.

51.2.3.2 Scheletro pronto con «skel.pl»

Da quanto è stato mostrato, si intende che la realizzazione di uno script con le specifiche di sostituzione per l'uso con '**sgmlspl**' non rappresenta un problema. Tuttavia, assieme alla documentazione di SGMLSpm si trova uno script speciale per '**sgmlspl**' che aiuta nella sua realizzazione iniziale. Si tratta di '**skel.pl**':

```
sgmlspl skel.pl < file_sp > scheletro_script_sgmlspl
```

Come si vede, si deve partire da un risultato generato da SP; da questo si ottiene uno scheletro per la realizzazione del proprio script di '**sgmlspl**'. In generale si tratta di qualcosa simile all'esempio seguente:

```
##############################################################################
# SGMLSPL script produced automatically by the script sgmlspl.pl
#
# Document Type: SGMLTEXI
# Edited by:
##############################################################################

use SGMLS;                      # Use the SGMLS package.
use SGMLS::Output;              # Use stack-based output.

#
# Document Handlers.
#
sgml('start', sub {});
sgml('end', sub {});
```

```
#
# Element Handlers.
#

# Element: SGMLTEXI
sgml('<SGMLTEXI>', "");
sgml('</SGMLTEXI>', "");

# Element: HEAD
sgml('<HEAD>', "");
sgml('</HEAD>', "");

# Element: ADMIN
sgml('<ADMIN>', "");
sgml('</ADMIN>', "");

…

#
# Default handlers (uncomment these if needed).  Right now, these are set
# up to gag on any unrecognised elements, sdata, processing-instructions,
# or entities.
#
# sgml('start_element',sub { die "Unknown element: " . $_[0]->name; });
# sgml('end_element','');
# sgml('cdata',sub { output $_[0]; });
# sgml('sdata',sub { die "Unknown SDATA: " . $_[0]; });
# sgml('re',"\n");
# sgml('pi',sub { die "Unknown processing instruction: " . $_[0]; });
# sgml('entity',sub { die "Unknown external entity: " . $_[0]->name; });
# sgml('start_subdoc',sub { die "Unknown subdoc entity: " . $_[0]->name; });
# sgml('end_subdoc','');
# sgml('conforming','');

1;
```

51.2.3.3 Ridirezione dell'output e altre sofisticazioni

Uno script per '**sgmlspl**' è in realtà uno script Perl. In questo senso si possono dichiarare variabili globali e funzioni aggiuntive. Questo consente di accumulare dei dati e di emetterli solo quando tutte le informazioni necessarie sono state ricevute, in un ordine differente rispetto alla struttura del sorgente SGML.

Per migliorare questa possibilità, è consentita la ridirezione del flusso generato attraverso la funzione ***output()***, in modo da poterlo ripescare al momento del bisogno. Prima di vedere come funziona questa cosa, si pensi a un problema tipico: si vuole accumulare in qualche modo l'informazione contenuta nell'elemento '**titolo**', in modo da poterla emettere nel momento appropriato.

In generale, si riesce a intercettare il marcatore di apertura e quello di chiusura dell'elemento, senza poter «afferrare» il testo contenuto. Più o meno nel modo seguente:

```
sgml('<TITOLO>', sub{
    output "\n\\section{";
});
sgml('</TITOLO>', sub{
    output "}";
});
```

Ma quel titolo potrebbe servire per qualche motivo. Ecco come si risolve il problema:

```
sgml('<TITOLO>', sub{
    output "\n\\section{";
    push_output('string');
});
sgml('</TITOLO>', sub{
    $titolo = pop_output;
    output "$titolo}";
});
```

Attraverso l'istruzione '**push_output('string')**' viene ridiretto temporaneamente tutto il flusso verso una stringa indefinita (viene chiarito meglio tra poco); successivamente viene prelevato il testo accumulato con l'istruzione '**pop_output**', inserendolo nella variabile '**$titolo**'. Infine, il testo accumulato viene anche emesso nuovamente attraverso la funzione ***output()***.

Ecco come si presenta la sintassi di queste due istruzioni:

```
push_output( tipo[, file] )
pop_output
```

In pratica, la funzione ***push_output()*** ridirige il flusso generato dalla funzione ***output()*** (che viene usata anche internamente a '**sgmlspl**'. Questo flusso può essere ridiretto verso oggetti differenti, identificati da una parola chiave che va indicata come primo argomento, come si vede nella tabella 51.121.

Tabella 51.121. Tipi di ridirezione della funzione ***push_output()***.

Tipo	Descrizione
'handle'	Ridirige verso un flusso aperto (*file handle* indicato nel secondo argomento.
'file'	Ridirige verso un file che viene creato per l'occasione.
'append'	Ridirige aggiungendo a un file esistente.
'pipe'	Ridirige inviando allo standard input del comando indicato.
'string'	Ridirige in un'area temporanea.
'nul'	Perde l'output.

La ridirezione verso un flusso di file già aperto, verso un file e verso un condotto è un concetto abbastanza intuitivo. In questi casi il secondo argomento indica il flusso, il file o il comando a cui si ridirige. Per esempio:

```
push_output( 'handle', MIO_FILE );
```

invia l'output verso il file già aperto con il nome '**MIO_FILE**';

```
push_output( 'file', "/tmp/pippo" );
```

genera il file '/tmp/pippo' e vi inserisce l'output;

```
push_output( 'append', "/tmp/pippo" );
```

accoda al file '/tmp/pippo';

```
push_output( 'pipe', "mail tizio" );
```

invia un messaggio di posta elettronica all'utente '**tizio**'.

Al contrario, il comando '**push_output('string')**' non prevede un secondo argomento e invia i dati in un'area indefinita, che può essere recuperata solo attraverso la funzione '**pop_output**'. La funzione '**pop_output**' serve in generale per concludere una ridirezione precedente, mentre quando si tratta in particolare di un flusso di output ridiretto verso questa area indefinita, restituisce quanto accumulato:

```
push_output('string');
…
…
$recupera = pop_output;
…
```

Infine, '**push_output('nul')**' serve a eliminare l'output senza poterlo recuperare.

51.2.3.4 Verifica sintattica secondo Perl

La verifica sintattica di uno script di '**sgmlspl**' può risultare difficile se non si usa un trucchetto: basta aggiungere la definizione di una funzione ***output()*** fittizia, come quella seguente:

```
#sub output {
#    local( $argument ) = $_[0];
#    print "$argument";
#}
```

L'esempio mostra delle righe commentate. Infatti, si tratta di inserire questa funzione fittizia solo nel momento in cui si vuole eseguire un'analisi attraverso Perl, nel modo seguente:

```
$ perl -c pippo.spec [Invio]
```

Qui, si intende che 'pippo.spec' sia lo script da controllare.

51.2.4 Esempio di un mini-sistema SGML

Il modo migliore per comprendere come si possono mettere insieme i vari tasselli di un sistema di composizione che parte dall'SGML, è quello di studiare un esempio elementare, che possa essere esteso facilmente. Si vuole arrivare a generare una trasformazione del sorgente SGML in LaTeX.

Quello che serve è: un DTD, che viene rappresentato dal file '`relazione.dtd`'; un catalogo, rappresentato dal file '`catalogo`'; una serie di file contenenti le entità standard ISO indispensabili e adatte a LaTeX, rappresentate dai file '`ISOlat1.tex`', '`ISOnum.tex`' e '`ISOdia.tex`'; e infine un file di rimpiazzo ASP, rappresentato dal file '`mappa.tex`', oppure un file di specifiche per '**`sgmlspl`**'.

51.2.4.1 Il DTD

Si vuole realizzare un tipo di documento molto semplice, adatto per scrivere delle relazioni banali, composte da un titolo, una data, un corpo più o meno lungo e da una o più firme.

```
<!ENTITY % ISOlat1 PUBLIC "ISO 8879:1986//ENTITIES Added Latin 1//EN">
%ISOlat1;

<!ENTITY % ISOdia PUBLIC "ISO 8879:1986//ENTITIES Diacritical Marks//EN">
%ISOdia;

<!ENTITY % ISOnum PUBLIC
        "ISO 8879:1986//ENTITIES Numeric and Special Graphic//EN">
%ISOnum;

<!ENTITY space " ">
<!ENTITY null "">

<!shortref mappaglobale
        "BB" space
        "&#RS;B" null
        "B&#RE;" space
        "&#RS;B&#RE;" null
        "&#RS;&#RE;" null
        "#" num
        "%" percnt
        "@" commat
        "[" lsqb
        "]" rsqb
        "^" circ
        "_" lowbar
        "{" lcub
        "|" verbar
        "}" rcub
        "~" tilde >

<!ELEMENT relazione     - - (titolo?, data, contenuto)>
<!ELEMENT titolo        - o (#PCDATA)>
<!ELEMENT data          - o (#PCDATA)>
<!ELEMENT contenuto     - o (paragrafo+, firma+)>
<!ELEMENT paragrafo     - o (#PCDATA)>
<!ELEMENT firma         - o (#PCDATA)>

<!usemap mappaglobale relazione>
```

Come si può osservare dall'esempio proposto, inizialmente vengono acquisite le entità standard, utilizzando un riferimento pubblico, secondo gli standard. Successivamente vengono definite delle entità aggiuntive e quindi una mappa di sostituzione (*shortref*).

Nella parte finale vengono definiti i vari elementi, a cominciare da quello che ha lo stesso nome del DTD, abbinando l'elemento più esterno all'unica mappa di sostituzione che sia stata definita.

51.2.4.2 Il catalogo

Il catalogo serve a individuare i file corrispondenti alle entità standard e al DTD stesso. Si tratta di poche righe (si osservi il fatto che non è stato definito un identificatore pubblico per il DTD, dal momento che si tratta di un lavoro poco importante).

```
PUBLIC "ISO 8879:1986//ENTITIES Added Latin 1//EN" "ISOlat1.tex"

PUBLIC "ISO 8879:1986//ENTITIES Diacritical Marks//EN" "ISOdia.tex"

PUBLIC "ISO 8879:1986//ENTITIES Numeric and Special Graphic//EN" "ISOnum.tex"

DOCTYPE "relazione"             "relazione.dtd"
```


51.2.4.3 Le entità standard

Le entità standard, come tali, si possono recuperare già pronte un po' dappertutto. Eventualmente si può porre il problema di dover modificare le stringhe corrispondenti per il tipo di elaborazione che si intende fare. Di seguito vengono mostrati integralmente i file delle entità utilizzati in questo esempio. È il caso di ricordare che le stringhe di sostituzione sono pensate per LaTeX.

```
<!-- Questa versione del file ISOlat1 è ridotta rispetto
     all'originale dello standard ISO 8879.
     Per la precisione, sono state tolte le entità che esistono
     già negli altri file mostrati.
-->
<!-- Character entity set. Typical invocation:
     <!ENTITY % ISOlat1 PUBLIC
       "ISO 8879:1986//ENTITIES Added Latin 1//EN">
     %ISOlat1;
-->
<!ENTITY aacute CDATA "\'a"--=small a, acute accent-->
<!ENTITY Aacute CDATA "\'A"--=capital A, acute accent-->
<!ENTITY acirc  CDATA "\^a"--=small a, circumflex accent-->
<!ENTITY Acirc  CDATA "\^A"--=capital A, circumflex accent-->
<!ENTITY agrave CDATA "\`a"--=small a, grave accent-->
<!ENTITY Agrave CDATA "\`A"--=capital A, grave accent-->
<!ENTITY aring  CDATA "\aa{}"--=small a, ring-->
<!ENTITY Aring  CDATA "\AA{}"--=capital A, ring-->
<!ENTITY atilde CDATA "\~a"--=small a, tilde-->
<!ENTITY Atilde CDATA "\~A"--=capital A, tilde-->
<!ENTITY auml   CDATA '\"a'--=small a, dieresis or umlaut mark-->
<!ENTITY Auml   CDATA '\"A'--=capital A, dieresis or umlaut mark-->
<!ENTITY aelig  CDATA "\ae{}"--=small ae diphthong (ligature)-->
<!ENTITY AElig  CDATA "\AE{}"--=capital AE diphthong (ligature)-->
<!ENTITY ccedil CDATA "\c c"--=small c, cedilla-->
<!ENTITY Ccedil CDATA "\c C"--=capital C, cedilla-->
<!ENTITY eth    CDATA "\dh{}"--=small eth, Icelandic-->
<!ENTITY ETH    CDATA "\DH{}"--=capital Eth, Icelandic-->
<!ENTITY eacute CDATA "\'e"--=small e, acute accent-->
<!ENTITY Eacute CDATA "\'E"--=capital E, acute accent-->
<!ENTITY ecirc  CDATA "\^e"--=small e, circumflex accent-->
<!ENTITY Ecirc  CDATA "\^E"--=capital E, circumflex accent-->
<!ENTITY egrave CDATA "\`e"--=small e, grave accent-->
<!ENTITY Egrave CDATA "\`E"--=capital E, grave accent-->
<!ENTITY euml   CDATA '\"e'--=small e, dieresis or umlaut mark-->
<!ENTITY Euml   CDATA '\"E'--=capital E, dieresis or umlaut mark-->
<!ENTITY iacute CDATA "\'\i{}"--=small i, acute accent-->
<!ENTITY Iacute CDATA "\'I"--=capital I, acute accent-->
<!ENTITY icirc  CDATA "\^\i{}"--=small i, circumflex accent-->
<!ENTITY Icirc  CDATA "\^I"--=capital I, circumflex accent-->
<!ENTITY igrave CDATA "\`\i{}"--=small i, grave accent-->
<!ENTITY Igrave CDATA "\`I"--=capital I, grave accent-->
<!ENTITY iuml   CDATA '\"\i{}'--=small i, dieresis or umlaut mark-->
<!ENTITY Iuml   CDATA '\"I'--=capital I, dieresis or umlaut mark-->
<!ENTITY ntilde CDATA "\~n"--=small n, tilde-->
<!ENTITY Ntilde CDATA "\~N"--=capital N, tilde-->
<!ENTITY oacute CDATA "\'o"--=small o, acute accent-->
<!ENTITY Oacute CDATA "\'O"--=capital O, acute accent-->
<!ENTITY ocirc  CDATA "\^o"--=small o, circumflex accent-->
<!ENTITY Ocirc  CDATA "\^O"--=capital O, circumflex accent-->
<!ENTITY ograve CDATA "\`o"--=small o, grave accent-->
<!ENTITY Ograve CDATA "\`O"--=capital O, grave accent-->
<!ENTITY oslash CDATA "\o{}"--=small o, slash-->
<!ENTITY Oslash CDATA "\O{}"--=capital O, slash-->
<!ENTITY otilde CDATA "\~o"--=small o, tilde-->
<!ENTITY Otilde CDATA "\~O"--=capital O, tilde-->
<!ENTITY ouml   CDATA '\"o'--=small o, dieresis or umlaut mark-->
<!ENTITY Ouml   CDATA '\"O'--=capital O, dieresis or umlaut mark-->
<!ENTITY szlig  CDATA "\ss{}"--=small sharp s, German (sz ligature)-->
<!ENTITY thorn  CDATA "\th{}"--=small thorn, Icelandic-->
<!ENTITY THORN  CDATA "\TH{}"--=capital THORN, Icelandic-->
<!ENTITY uacute CDATA "\'u"--=small u, acute accent-->
<!ENTITY Uacute CDATA "\'U"--=capital U, acute accent-->
<!ENTITY ucirc  CDATA "\^u"--=small u, circumflex accent-->
<!ENTITY Ucirc  CDATA "\^U"--=capital U, circumflex accent-->
<!ENTITY ugrave CDATA "\`u"--=small u, grave accent-->
<!ENTITY Ugrave CDATA "\`U"--=capital U, grave accent-->
<!ENTITY uuml   CDATA '\"u'--=small u, dieresis or umlaut mark-->
<!ENTITY Uuml   CDATA '\"U'--=capital U, dieresis or umlaut mark-->
<!ENTITY yacute CDATA "\'y"--=small y, acute accent-->
<!ENTITY Yacute CDATA "\'Y"--=capital Y, acute accent-->
<!ENTITY yuml   CDATA '\"y'--=small y, dieresis or umlaut mark-->
```

```
<!-- (C) International Organization for Standardization 1986
     Permission to copy in any form is granted for use with
     conforming SGML systems and applications as defined in
     ISO 8879, provided this notice is included in all copies.
-->
<!-- Character entity set. Typical invocation:
     <!ENTITY % ISOnum PUBLIC
       "ISO 8879:1986//ENTITIES Numeric and Special Graphic//EN">
     %ISOnum;
-->
```

```
<!ENTITY half   CDATA "$\scriptstyle{1\over2}$"--=fraction one-half-->
<!ENTITY frac12 CDATA "\sfrac1/2"--=fraction one-half-->
<!ENTITY frac14 CDATA "\sfrac1/4"--=fraction one-quarter-->
<!ENTITY frac34 CDATA "\sfrac3/4"--=fraction three-quarters-->
<!ENTITY frac18 CDATA "\sfrac1/8"--=fraction one-eighth-->
<!ENTITY frac38 CDATA "\sfrac3/8"--=fraction three-eighths-->
<!ENTITY frac58 CDATA "\sfrac5/8"--=fraction five-eighths-->
<!ENTITY frac78 CDATA "\sfrac7/8"--=fraction seven-eighths-->

<!ENTITY sup1   CDATA "$^1$"--=superscript one-->
<!ENTITY sup2   CDATA "$^2$"--=superscript two-->
<!ENTITY sup3   CDATA "$^3$"--=superscript three-->

<!ENTITY plus   CDATA "$+$"--=plus sign B:-- >
<!ENTITY plusmn CDATA "$\pm$"--/pm B: =plus-or-minus sign-->
<!ENTITY lt     CDATA "$<$"--=less-than sign R:-->
<!ENTITY equals CDATA "$=$"--=equals sign R:-->
<!ENTITY gt     CDATA "$>$"--=greater-than sign R:-->
<!ENTITY divide CDATA "$\div$"--/div B: =divide sign-->
<!ENTITY times  CDATA "$\times$"--/times B: =multiply sign-->

<!ENTITY curren CDATA "\{curren\}"--=general currency sign-->
<!ENTITY pound  CDATA "\pounds{}"--=pound sign-->
<!ENTITY dollar CDATA "\$"--=dollar sign-->
<!ENTITY cent   CDATA "\cent{}"--=cent sign-->
<!ENTITY yen    CDATA "\{yen\}"--/yen =yen sign-->

<!ENTITY num    CDATA "\#"--=number sign-->
<!ENTITY percnt CDATA "\%"--=percent sign-->
<!ENTITY amp    CDATA "\&"--=ampersand-->
<!ENTITY ast    CDATA "*"--/ast B: =asterisk-->
<!ENTITY commat CDATA "@"--=commercial at-->
<!ENTITY lsqb   CDATA "["--/lbrack O: =left square bracket-->
<!ENTITY bsol   CDATA "$\backslash$"--/backslash =reverse solidus-->
<!ENTITY rsqb   CDATA "]"--/rbrack C: =right square bracket-->
<!ENTITY lcub   CDATA "$\{$"--/lbrace O: =left curly bracket-->
<!ENTITY horbar CDATA "{--}"--=horizontal bar-->
<!ENTITY verbar CDATA "$|$"--/vert =vertical bar-->
<!ENTITY rcub   CDATA "$\}$"--/rbrace C: =right curly bracket-->
<!ENTITY micro  CDATA "$\mu$"--=micro sign-->
<!ENTITY ohm    CDATA "$\Omega$"--=ohm sign-->
<!ENTITY deg    CDATA "$^\circ$"--=degree sign-->
<!ENTITY ordm   CDATA "\{ordm\}"--=ordinal indicator, masculine-->
<!ENTITY ordf   CDATA "\{ordf\}"--=ordinal indicator, feminine-->
<!ENTITY sect   CDATA "\S{}"--=section sign-->
<!ENTITY para   CDATA "\P{}"--=pilcrow (paragraph sign)-->
<!ENTITY middot CDATA "$\cdot$"--/centerdot B: =middle dot-->
<!ENTITY larr   CDATA "$\leftarrow$"--/leftarrow /gets A: =leftward arrow-->
<!ENTITY rarr   CDATA "$\rightarrow$"--/rightarrow /to A: =rightward arrow-->
<!ENTITY uarr   CDATA "$\uparrow$"--/uparrow A: =upward arrow-->
<!ENTITY darr   CDATA "$\downarrow$"--/downarrow A: =downward arrow-->
<!ENTITY copy   CDATA "\copyright{}"--=copyright sign-->
<!ENTITY reg    CDATA "\rcircle{}"--/circledR =registered sign-->
<!ENTITY trade  CDATA "(TM)"--=trade mark sign-->
<!ENTITY brvbar CDATA "\{brvbar\}"--=broken (vertical) bar-->
<!ENTITY not    CDATA "$\neg$"--/neg /lnot =not sign-->
<!ENTITY sung   CDATA "\{sung\}"--=music note (sung text sign)-->

<!ENTITY excl   CDATA "!"--=exclamation mark-->
<!ENTITY iexcl  CDATA "{!'}"--=inverted exclamation mark-->
<!ENTITY quot   CDATA '{\tt\char'\"}'--=quotation mark-->
<!ENTITY apos   CDATA "'"--=apostrophe-->
<!ENTITY lpar   CDATA "("--O: =left parenthesis-->
<!ENTITY rpar   CDATA ")"--C: =right parenthesis-->
<!ENTITY comma  CDATA ","--P: =comma-->
<!ENTITY lowbar CDATA "\_"--=low line-->
<!ENTITY hyphen CDATA "-"--=hyphen-->
<!ENTITY period CDATA "."--=full stop, period-->
<!ENTITY sol    CDATA "/"--=solidus-->
<!ENTITY colon  CDATA ":"--/colon P:-->
<!ENTITY semi   CDATA ";"--=semicolon P:-->
<!ENTITY quest  CDATA "?"--=question mark-->
<!ENTITY iquest CDATA "{?'}"--=inverted question mark-->
<!ENTITY laquo  CDATA "\guillemotleft{}"--=angle quotation mark, left-->
<!ENTITY raquo  CDATA "\guillemotright{}"--=angle quotation mark, right-->
<!ENTITY lsquo  CDATA "{'}"--=single quotation mark, left-->
<!ENTITY rsquo  CDATA "{'}"--=single quotation mark, right-->
<!ENTITY ldquo  CDATA "{''}"--=double quotation mark, left-->
<!ENTITY rdquo  CDATA "{''}"--=double quotation mark, right-->
<!ENTITY nbsp   CDATA "~"--=no break (required) space-->
<!ENTITY shy    CDATA "\-"--=soft hyphen-->
```



```
<!-- (C) International Organization for Standardization 1986
     Permission to copy in any form is granted for use with
     conforming SGML systems and applications as defined in
     ISO 8879, provided this notice is included in all copies.
-->
<!-- Character entity set. Typical invocation:
     <!ENTITY % ISOdia PUBLIC
       "ISO 8879:1986//ENTITIES Diacritical Marks//EN">
     %ISOdia;
-->
<!ENTITY acute  CDATA "\'"--=acute accent-->
<!ENTITY breve  CDATA "\u{"--=breve-->
<!ENTITY caron  CDATA "\{caron\}"--=caron-->
<!ENTITY cedil  CDATA "\c{"--=cedilla-->
<!ENTITY circ   CDATA "\^{}"--=circumflex accent-->
<!ENTITY dblac  CDATA "\{dblac\}"--=double acute accent-->
<!ENTITY die    CDATA '\"'--=dieresis-->
<!ENTITY dot    CDATA "\."--=dot above-->
<!ENTITY grave  CDATA "\`"--=grave accent-->
<!ENTITY macr   CDATA "\="--=macron-->
<!ENTITY ogon   CDATA "\{ogon\}"--=ogonek-->
<!ENTITY ring   CDATA "\accent23"--=ring-->
<!ENTITY tilde  CDATA "\~{}"--=tilde-->
<!ENTITY uml    CDATA '\"'--=umlaut mark-->
```

51.2.4.4 Rimpiazzo ASP

L'ultimo componente necessario è il file di rimpiazzo ASP per '**sgmlsasp**', oppure il file delle specifiche per '**sgmlspl**'. Vengono mostrati entrambi.

```
%
% mappa.tex
%
<relazione>     +       "\\documentclass{article}\n"
                        "\\begin{document}"             +

</relazione>    +       "\\end{document}"       +

<titolo>        +       "\n\\section{"
</titolo>               "}"             +

<data>          +       "\n"
</data>                 ""      +

<contenuto>
</contenuto>

<paragrafo>     +       "\n"
</paragrafo>            ""      +

<firma>         +       "\n"
</firma>                ""      +
```

```
#
# latex.spec
#
sgml( '<RELAZIONE>', sub {
    output "\n\\documentclass{article}\n";
    output "\\begin{document}";
});
sgml( '</RELAZIONE>', "\n\\end{document}\n" );

sgml( '<TITOLO>', "\n\n\\section{" );
sgml( '</TITOLO>', "}\n" );

sgml( '<DATA>', "\n\n" );
sgml( '</DATA>', "" );

sgml( '<CONTENUTO>', "" );
sgml( '</CONTENUTO>', "" );

sgml( '<PARAGRAFO>', "\n\n" );
sgml( '</PARAGRAFO>', "" );

sgml( '<FIRMA>', "\n\n" );
sgml( '</FIRMA>', "" );
```

51.2.4.5 I documenti SGML

Quello che segue è un esempio di documento SGML adatto al DTD e al catalogo che è stato definito sopra.

```
<!doctype relazione SYSTEM>
```

```
<relazione>
<titolo>Relazione introduttiva su SGML</titolo>

<data>31/12/1999</data>

<contenuto>

<paragrafo>SGML sta per Standard Generalized Markup
Language. bla bla bla... Perch&eacute;,... cos&igrave;...

<paragrafo>Bla, bla, bla....

<firma>Pinco Pallino</firma>

</contenuto>
</relazione>
```

51.2.4.6 I comandi necessari

Una volta scritto un testo SGML, la prima cosa da fare è la verifica di coerenza in base al DTD. Se il file da controllare fosse '`relazione.sgml`', si dovrebbe utilizzare il comando seguente (si presume che il file del catalogo sia collocato nella directory corrente).

```
$ nsgmls -s -c ./catalogo < relazione.sgml [Invio]
```

Una volta corretti gli errori, si può passare direttamente alla trasformazione in LaTeX, ma se lo si desidera, si può osservare l'output generato da '**`nsgmls`**'.

```
$ nsgmls -c ./catalogo < relazione.sgml [Invio]
```

Se si tratta dell'esempio di documento mostrato in precedenza, il risultato dovrebbe essere il seguente (una riga molto lunga appare spezzata per motivi tipografici).

```
(RELAZIONE
(TITOLO
-Relazione introduttiva su SGML
)TITOLO
(DATA
-31/12/1999
)DATA
(CONTENUTO
(PARAGRAFO
SGML sta per Standard Generalized Markup Language.\nbla ↩
↪bla bla... Perch\\'e,... cos\\'\\i{}...
)PARAGRAFO
(PARAGRAFO
-Bla, bla, bla....
)PARAGRAFO
(FIRMA
-Pinco Pallino
)FIRMA
)CONTENUTO
)RELAZIONE
C
```

Vale la pena di concentrare l'attenzione proprio sulla riga più lunga, nella quale si può osservare l'effetto delle sostituzioni delle entità standard, secondo le esigenze di LaTeX.:

```
(PARAGRAFO
SGML sta per Standard Generalized Markup Language.\nbla ↩
↪bla bla... Perch\\'e,... cos\\'\\i{}...
)PARAGRAFO
```

Il comando completo per ottenere una trasformazione in LaTeX, secondo il contenuto del file di rimpiazzo ('`mappa.tex`'), è il seguente:

```
$ cat relazione.sgml | nsgmls -c ./catalogo ↩
↪| sgmlsasp mappa.tex [Invio]
```

Ovvero, nel caso si utilizzi '**`sgmlspl`**':

```
$ cat relazione.sgml | nsgmls -c ./catalogo ↩
↪| sgmlspl latex.spec [Invio]
```

Il risultato ottenuto attraverso lo standard output, che andrebbe ridiretto opportunamente, potrebbe apparire come quello che segue.


```
\documentclass{article}
\begin{document}

\section{Relazione introduttiva su SGML}

31/12/1999

SGML sta per Standard Generalized Markup Language.
bla bla bla... Perch\'e,... cos\'\i{}...

Bla, bla, bla....

Pinco Pallino
\end{document}
```

51.2.5 Lo scalino successivo

Una volta capito come si possono utilizzare gli strumenti SGML comuni, si pongono subito due tipi di problemi: la gestione simultanea di più sistemi di composizione e l'astrazione dal problema della rappresentazione dei simboli che in uno qualunque dei sistemi di composizione richiederebbero codici speciali. Vengono analizzati questi due problemi separatamente.

51.2.5.1 Gestione simultanea di più sistemi di composizione

Quando si organizza un DTD allo scopo di costruire un sistema SGML per la composizione finale in più formati (PostScript, HTML ed eventualmente altro ancora), occorre definire quali siano gli obiettivi, stabilendo così anche i limiti che si devono imporre nel DTD (se si pretende di generare anche un risultato in forma di file di testo puro e semplice, le immagini potrebbero essere inserite nel documento solo in forma di «arte ASCII»).

Dopo il progetto del DTD e del modo in cui devono essere trasformati i vari elementi nelle diverse forme di composizione, si pone un ostacolo un po' fastidioso: le entità generali. Dal momento che queste dovrebbero essere definite in modo differente a seconda del tipo di composizione che si vuole ottenere, si rischia di dover gestire altrettanti cataloghi, dovendo fare riferimento a file differenti.

In un sistema SGML ben ordinato, ci dovrebbe essere un solo catalogo e il problema della distinzione delle entità generali si può ottenere attraverso l'uso delle sezioni marcate. Infatti, dal momento che i file delle entità esterne sono parte del DTD, si possono indicare anche altre istruzioni SGML oltre a quelle di definizione delle entità generali. Quello che segue è un estratto semplificato e abbreviato dal file delle entità esterne utilizzato da ALtools (un vecchio sistema di composizione di *Appunti Linux*).

```
<![CDATA[
<!ENTITY % EntitaASCII8 "IGNORE">
<!ENTITY % EntitaLaTeX "IGNORE">
<!ENTITY % EntitaHTML "IGNORE">

<![ %EntitaASCII8 [
    <!ENTITY excl   CDATA "!"-- exclamation mark -->
    <!ENTITY quot   CDATA '"'-- quotation mark -->
    <!ENTITY num    CDATA "#"-- number sign -->
    …
    …
]]>

<![ %EntitaLaTeX [
    <!ENTITY excl   CDATA "!"-- exclamation mark -->
    <!ENTITY quot   CDATA '{\tt\char'\"}'-- quotation mark -->
    <!ENTITY num    CDATA "\#"-- number sign -->
    …
    …
]]>

<![ %EntitaHTML [
    <!ENTITY excl   CDATA "!"-- exclamation mark -->
    <!ENTITY quot   CDATA '"'-- quotation mark -->
    <!ENTITY num    CDATA "#"-- number sign -->
    …
    …
]]>
```

Nella parte iniziale vengono dichiarate le entità parametriche ‘**EntitaASCII8**’, ‘**EntitaLaTeX**’ e ‘**EntitaHTML**’, tutte con la stringa ‘**IGNORE**’. In questo modo, in condizioni normali, nessuna delle istruzioni di definizioni delle entità generali verrebbe presa in considerazione. Per selezionare un gruppo soltanto, basterebbe che l’entità parametrica giusta contenesse la stringa ‘**INCLUDE**’. Per farlo si interviene direttamente nella riga di comando di ‘**nsgmls**’ (SP):

```
cat file_sgml | nsgmls -c catalogo -ientità_parametrica | ...
```

In pratica, con l’opzione ‘**-i**’ di ‘**nsgmls**’, si fa in modo di introdurre una dichiarazione del tipo

```
<!ENTITY % entità_parametrica "INCLUDE">
```

e questa prevale automaticamente su qualunque altra dichiarazione analoga (della stessa entità parametrica) in qualunque altra parte del DTD.

Per tornare all’esempio mostrato del file delle entità generali, si potrebbero selezionare le entità riferite alla trasformazione in LaTeX con un comando simile a quello seguente:

```
$ cat mio_file.sgml ↩
↪| nsgmls -c ./catalogo -iEntitaLaTeX | ... [Invio]
```

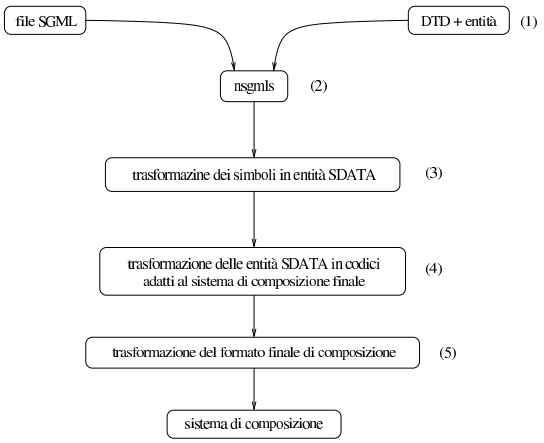
51.2.5.2 Insieme di caratteri

Attraverso le entità generali che si definiscono, è possibile fare in modo che il sistema di composizione finale riceva i codici adatti per tutti i simboli «strani» che si vogliono poter inserire. Tuttavia, spesso si vorrebbe poter scrivere liberamente utilizzando il minor numero possibile di macro ‘**&…;**’. Per la precisione, il minimo in assoluto è quello che richiede l’SGML stesso: occorre proteggere i simboli ‘**&**’, ‘**>**’ e ‘**<**’ (‘**&**’, ‘**>**’, ‘**<**’). Tutto il resto, non dà alcun fastidio all’analizzatore SGML, però i programmi di composizione potrebbero avere dei problemi differenti.

Anche senza uscire dai 7 bit dell’ASCII tradizionali, se si scrive qualcosa per LaTeX, non si possono usare direttamente caratteri normalissimi come ‘**#**’, ‘****’, ‘**$**’ e altri.

Per risolvere questo problema una volta per tutte, si utilizza una tecnica che impone una rielaborazione intermedia del risultato generato da SP dall’analisi del sorgente SGML. Questa tecnica si basa sull’uso di entità generali di tipo ‘**SDATA**’. Quando queste vengono sostituite dallo stesso analizzatore SGML, appaiono delimitate dalla sequenza ‘**\|**’, cosa che ne facilita l’individuazione da parte di un programma di rielaborazione.

Figura 51.140. Passaggi per risolvere il problema dell’insieme dei caratteri.



In questo modo si perde il vantaggio di lasciare fare all’SGML la sostituzione delle entità, però ci si può limitare a intervenire solo dove serve.

Quando si decide di intraprendere questo tipo di approccio, occorre ricordare che l’elaborazione dell’output di ‘**nsgmls**’ deve evitare di intervenire negli elementi «letterali», ovvero quelli che anche nel sistema di composizione finale vengono presi e riprodotti tali e quali.

La descrizione seguente fa riferimento alla figura 51.140.

1. Le entità riferite ai simboli che possono creare problemi vengono definite in una qualche forma simbolica specificando il tipo ‘**SDATA**’. Per esempio, il carattere ‘**#**’ potrebbe essere definito nel modo standard:

   ```
   <!ENTITY num    SDATA "[num   ]"-- number sign -->
   ```

2. ‘**sgmls**’ elabora il file SGML e sostituisce le entità. Quando incontra per esempio la macro ‘**#**’, la trasforma in ‘**\|[num]\|**’.
3. Un programma di elaborazione successivo, quando incontra per esempio il carattere ‘**#**’, lo trasforma in quello che sarebbe stato generato se fosse stata usata la macro ‘**#**’; in pratica lo trasforma in ‘**\|[num]\|**’.
4. A questo punto, i simboli come ‘**#**’ che potrebbero provocare problemi sono stati trasformati tutti nella forma ‘**\|[num]\|**’. Quindi, un programma si deve occupare di trasformarli nel modo adatto al sistema di composizione a cui si devono poi dare in pasto i dati. Nel caso di LaTeX, la stringa ‘**\|[num]\|**’ viene sostituita con ‘**\\#**’. Nel risultato finale, LaTeX richiede solo la stringa ‘**\#**’, ma fino a che si resta nell’ambito del risultato generato da ‘**nsgmls**’, le barre oblique inverse devono essere raddoppiate.
5. Attraverso ‘**sgmlsasp**’, oppure ‘**sgmlspl**’, si genera il risultato finale da passare al sistema di composizione.

51.2.6 Organizzazione degli strumenti SGML in una distribuzione GNU/Linux

Non sempre le distribuzioni GNU/Linux si occupano di organizzare gli strumenti SGML, mentre questo sarebbe molto importante per tutti gli sviluppatori di programmi riferiti a questo standard e a quelli derivati. A questo proposito, vale la pena di osservare la distribuzione Debian che mette in pratica alcune buone idee.

Il problema fondamentale sta nello stabilire la collocazione dei DTD e dei file delle entità generali relative. Infine, si tratta di definire un catalogo unico per tutti questi DTD e per i file delle entità. I file dei DTD vengono collocati nella directory ‘`/usr/share/sgml/dtd/`’, mentre quelli delle entità si trovano nella directory ‘`/usr/share/sgml/entities/`’. A questo punto, per facilitare l’indicazione dei file nel catalogo, questo dovrebbe trovarsi opportunamente nella directory ‘`/etc/sgml/`’, con il nome ‘`catalog`’; così il file del catalogo può essere aggiornato senza interferire con la gerarchia ‘`/usr/`’ che deve poter essere innestata in sola lettura.

Avendo organizzato tutto in questo modo, ogni volta che si installa un nuovo pacchetto di strumenti SGML, questo dovrebbe provvedere ad aggiungere nel catalogo standard tutte le dichiarazioni che lo riguardano.

La base di questa struttura nella distribuzione Debian è costituita dai pacchetti ‘`sgml-base_*.deb`’ e ‘`sgml-data_*.deb`’.

51.2.7 perlSGML: analisi di un DTD

Quando si realizza un DTD per qualche scopo, potrebbe essere importante disporre di strumenti adatti alla sua analisi, per verificare la sua coerenza con l’obiettivo che ci si pone. Sono importanti a questo proposito i programmi di servizio del pacchetto perlSGML. Qui ne vengono mostrati solo alcuni.

In generale, per fare in modo che questi programmi di analisi funzionino correttamente, è opportuno che la directory corrente nel momento in cui si avviano corrisponda a quella in cui si trova il catalogo, in maniera tale che poi da lì, possa trovare le entità che fossero state collocate eventualmente in un file esterno. Se poi il file del catalogo non si chiama '`catalog`', occorre usare l'opzione opportuna per indicare il nome corretto.

Il primo programma che qui viene preso in considerazione a proposito di perlSGML è '**`dtd2html`**':

```
dtd2html [opzioni] file_dtd...
```

'**`dtd2html`**' genera un rapporto sui DTD elencati alla fine degli argomenti, in forma di ipertesto HTML.

Tabella 51.142. Alcune opzioni.

Opzione	Descrizione
`-help`	Emette un riepilogo dell'utilizzo del programma.
`-catalog` *catalogo*	Permette di indicare il nome del file contenente il catalogo SGML. In mancanza di questa opzione, viene cercato il file '`catalog`' nella directory corrente.
`-outdir` *directory*	Permette di specificare una directory diversa da quella corrente, nella quale vanno generate le pagine HTML.
`-ents`	Fa in modo che venga aggiunta una pagina HTML con l'elenco delle entità dichiarate nel corpo principale del DTD.
`-tree`	Fa in modo che venga aggiunta una pagina HTML con l'albero degli elementi SGML collegati tra loro in base alle dipendenze relative.

Segue la descrizione di alcuni esempi.

- `$` **`dtd2html dtd/mio.dtd`** [*Invio*]

  Analizza il file '`./dtd/mio.dtd`' utilizzando il catalogo '`./catalog`' e generando i file HTML nella directory corrente.

- `$` **`dtd2html -catalog catalogo dtd/mio.dtd`** [*Invio*]

  Come nell'esempio precedente, specificando che il catalogo è contenuto nel file '`./catalogo`'.

- `$` **`dtd2html -catalog catalogo -outdir /tmp dtd/mio.dtd`** [*Invio*]

  Come nell'esempio precedente, richiedendo che i file HTML siano creati nella directory '`/tmp/`'.

- `$` **`dtd2html -catalog catalogo -outdir /tmp -ents dtd/mio.dtd`** [*Invio*]

  Come nell'esempio precedente, richiedendo anche la generazione di una pagina dedicata alle entità dichiarate nel DTD.

- `$` **`dtd2html -catalog catalogo -outdir /tmp -ents -tree dtd/mio.dtd`** [*Invio*]

  Come nell'esempio precedente, richiedendo anche la generazione di una pagina contenente l'albero degli elementi.

Il programma '**`dtddiff`**' permette di confrontare due DTD, per conoscere le differenze di contenuto tra i due. Il risultato viene emesso attraverso lo standard output:

```
dtddiff [opzioni] file_dtd file_dtd
```

Tabella 51.143. Alcune opzioni.

Opzione	Descrizione
`-help`	Emette un riepilogo dell'utilizzo del programma.
`-catalog` *catalogo*	Permette di indicare il nome del file contenente il catalogo SGML. In mancanza di questa opzione, viene cercato il file '`catalog`' nella directory corrente.

Segue la descrizione di alcuni esempi.

- `$` **`dtddiff -catalog catalogo dtd/mio.dtd dtd2/mio.dtd`** [*Invio*]

  Confronta i DTD '`./dtd/mio.dtd`' e '`./dtd/mio2.dtd`', utilizzando il catalogo '`./catalogo`'.

51.3 Dichiarazione SGML

Fino a questo punto è stata ignorata la dichiarazione SGML, che in generale non dovrebbe essere un problema per l'utilizzatore, ma rappresenta pur sempre un elemento determinante per la comprensione della filosofia di questo linguaggio.

La dichiarazione SGML è qualcosa che viene prima del DTD; serve a definire la forma del sorgente e alcune caratteristiche del linguaggio utilizzato. Attraverso la dichiarazione si possono modificare molti comportamenti convenzionali, facendo anche cambiare aspetto notevolmente al linguaggio stesso. Tutto quello che è stato descritto di SGML fino a questo punto, fa affidamento sulla dichiarazione SGML raccomandata, ma volendo si potrebbero cambiare molte cose. Per fare un esempio pratico, XML può essere inteso come un modo di utilizzare SGML in base a una dichiarazione particolare, realizzata per le esigenze specifiche della pubblicazione di documentazione attraverso la rete.

La dichiarazione SGML si fa generalmente in un file apposito; tutte le direttive sono contenute all'interno di un'istruzione sola del tipo seguente:

```
<!SGML "ISO 8879:1986"
…
…
…
>
```

In pratica, nel modello mostrato, le direttive occupano il posto dei puntini di sospensione.

Si osservi che lo standard originale ISO prevedeva la definizione '**`8879-1986`**', che successivamente è stata modificata nel modo mostrato, ovvero '**`8879:1986`**'. Lo stesso ragionamento vale per gli altri standard ISO che prevedono l'indicazione dell'anno.

Esiste una variante recente allo standard ISO 8879:1996 e precisamente si tratta di cambiamenti pensati per facilitare la comunicazione attraverso la rete. La stringa che fa riferimento a questo standard esteso è:

```
"ISO 8879:1986 (WWW)"
```

La si ritrova in particolare nella dichiarazione dell'HTML 4.∗ e nell'XML.

In questo capitolo vengono mostrate solo alcune direttive che possono essere utili per capire il senso della dichiarazione SGML. Per approfondire lo studio di questo linguaggio, bisogna procurarsi la documentazione originale ISO.

51.3.1 Codifica

La codifica dei caratteri utilizzata nel sorgente SGML non può essere ignorata, soprattutto perché alcuni codici hanno significati speciali che vanno oltre il carattere vero e proprio. Le direttive riferite alla codifica del sorgente iniziano con la parola chiave '**`CHARSET`**' che delimita la sezione relativa:

```
CHARSET
    definizione_riferita_all'insieme_di_caratteri
    …
```

In generale, si inizia con la definizione di un insieme standard di riferimento, attraverso l'uso di un identificatore standard:

```
BASESET insieme_di_caratteri
```

L'identificatore che definisce lo standard è normalmente una stringa abbastanza dettagliata. L'esempio seguente definisce l'insieme di partenza corrispondente all'ISO 646:1983, ovvero all'ASCII tradizionale:

```
BASESET  "ISO 646:1983//CHARSET
         International Reference Version (IRV)//ESC 2/5 4/0"
```

La direttiva appare su due righe, ma si tratta solo di una possibilità e non di una necessità, tanto che in alcuni casi la si può vedere anche distribuita su tre righe. Dopo la definizione dell'insieme di partenza, si può descrivere nel dettaglio l'utilizzo e la conversione dei codici corrispondenti ai caratteri:

```
DESCSET
    inizio  quantità  {corrispondenza | UNUSED}
    …
```

Si osservi l'esempio:

```
    DESCSET
          0  9 UNUSED
          9  2  9
         11  2 UNUSED
         13  1 13
         14 18 UNUSED
         32 95 32
        127  1 UNUSED
```

Il primo numero indica il codice corrispondente al carattere iniziale di un raggruppamento composto da una sequenza di ***n*** caratteri; il secondo valore indica una quantità di caratteri che possono essere ignorati oppure anche trasformati, partendo dal codice rappresentato dal terzo valore.

Nell'esempio, i codici che vanno da 0 a 8, in decimale (nel senso che si tratta di valori espressi in base 10), non sono utilizzati; inoltre i codici da 9 a 10 vengono convertiti con il codice 9 e seguenti (in pratica non vengono convertiti affatto). In sostanza, ciò che mostra l'esempio non ha lo scopo di convertire alcunché, ma solo di filtrare codici inutili: vengono lasciati passare i caratteri grafici, a partire dallo spazio, oltre a *<HT>*, *<LF>* e *<CR>*. Volendo esprimere la cosa in modo più esplicito, si possono usare anche dei commenti descrittivi:

```
    DESCSET
          0  9 UNUSED
          9  1  9       -- HT --
         10  1 10       -- LF --
         11  2 UNUSED
         13  1 13       -- CR --
         14 18 UNUSED
         32 95 32       -- SP e altri caratteri grafici --
        127  1 UNUSED
```

La sequenza di direttive '**BASESET**' e '**DESCSET**' può anche essere ripetuta, quando dopo l'ASCII normale, i primi 7 bit, si vuole fare riferimento a qualcosa di più. Per esempio, la dichiarazione relativa alla codifica dell'HTML 3.2, si presenta come si vede di seguito:

```
CHARSET
     BASESET  "ISO 646:1983//CHARSET
               International Reference Version
               (IRV)//ESC 2/5 4/0"
     DESCSET  0   9   UNUSED
              9   2   9
              11  2   UNUSED
              13  1   13
              14  18  UNUSED
              32  95  32
              127 1   UNUSED
     BASESET  "ISO Registration Number 100//CHARSET
               ECMA-94 Right Part of
               Latin Alphabet Nr. 1//ESC 2/13 4/1"
     DESCSET  128  32   UNUSED
              160  96    32
```


Rispetto a quanto già visto si aggiunge il riferimento allo standard ISO 8859-1 (Latin 1). Si può vedere che vengono esclusi i primi 32 codici a partire dal numero 128, che non contengono simboli grafici utili.

51.3.2 Capacità

Per qualche ragione storica, che ormai non avrebbe più motivo di sussistere, è prevista una sezione attraverso la quale si definisce la capacità elaborativa dell'analizzatore SGML. Si tratta di stabilire dei limiti di spazio per la gestione di una serie di informazioni. In generale, non dovrebbe essere determinante la dimensione da dare ai vari attributi riferiti a questa capacità; tuttavia, si tratta di un'indicazione che rimane, per la quale si fa riferimento allo standard, oppure si indica semplicemente che non ci sono limiti. Nel primo caso si indica,

```
CAPACITY PUBLIC  "ISO 8879:1986//CAPACITY Reference//EN"
```

nel secondo soltanto

```
CAPACITY NONE
```

A titolo di esempio si mostra anche la direttiva relativa riferita all'HTML 3.2 e 4:

```
CAPACITY        SGMLREF
                TOTALCAP        150000
                GRPCAP          150000
                ENTCAP          150000
```

Si osservi la parola chiave '**SGMLREF**' che può essere usata anche altrove. Rappresenta il riferimento ai valori predefiniti SGML, prima di modificarli o integrarli con le richieste successive.

51.3.3 Ambito

La sintassi del linguaggio SGML può essere alterata in parte, attraverso una serie di direttive descritte nella prossima sezione. L'ambito della definizione della sintassi SGML può essere controllato attraverso la direttiva '**SCOPE**':

```
SCOPE DOCUMENT | INSTANCE
```

La direttiva '**SCOPE DOCUMENT**' indica che la sintassi si applica sia al DTD, sia al sorgente SGML; nell'altro caso, '**SCOPE INSTANCE**' si riferisce solo al sorgente, mentre il DTD va interpretato in base alla sintassi standard predefinita (la ***sintassi concreta di riferimento***).

Di solito si usa la direttiva '**SCOPE DOCUMENT**'.

51.3.4 Sintassi concreta

La ***sintassi concreta*** è ciò che definisce i delimitatori dei marcatori SGML, il ruolo dei codici di controllo e altri dettagli riferiti alla sintassi SGML. In particolare si parla di sintassi concreta di riferimento quando si vuole indicare quella predefinita, ovvero quella a cui si fa riferimento di solito. Le direttive che compongono la definizione della sintassi concreta sono introdotte dalla sezione '**SYNTAX**', a cui spesso segue la stringa di un identificatore pubblico, per richiamare inizialmente una serie di caratteristiche standard che poi vengono alterate o integrate dalle direttive successive:

```
SYNTAX   PUBLIC  "ISO 8879:1986//SYNTAX Reference//EN"
```

51.3.4.1 Caratteri da evitare

La prima cosa che si specifica all'interno della dichiarazione della sintassi concreta è l'elenco dei numeri decimali corrispondenti ai codici, o caratteri, che non devono essere usati nel testo del sorgente. Questi non vengono passati all'applicazione successiva dall'analizzatore SGML. All'interno dei codici esclusi in questo modo ci possono essere comunque simboli o caratteri di controllo che servono in altri ambiti, come viene mostrato in seguito.

La codifica a cui si fa riferimento, non è quella ottenuta dopo la trasformazione con la direttiva '**DESCSET**' della sezione '**CHARSET**', ma quella della stessa direttiva della sezione '**SYNTAX**', come viene descritto tra poco.

La direttiva in questione è molto semplice; spesso, quando si tratta dell'ASCII, si utilizza direttamente l'esempio seguente:

```
SHUNCHAR CONTROLS 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
         16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
         127
```

51.3.4.2 Codifica nell'ambito della sintassi concreta

Nell'ambito della definizione della sintassi concreta, è necessario specificare nuovamente la codifica di partenza e la conversione eventuale. Tutto procede esattamente come è già stato visto in precedenza, nella sezione '**CHARSET**', con la differenza che in generale si preferisce lasciare tutto come si trova:

```
BASESET  "ISO 646:1983//CHARSET
         International Reference Version
         (IRV)//ESC 2/5 4/0"
DESCSET  0 128 0
```

L'esempio si riferisce al caso in cui si utilizzi solo l'ASCII. Comunque, si può osservare che la direttiva '**DESCSET**' non esclude alcunché e non trasforma alcun carattere.

51.3.4.3 Codici con funzioni speciali

Si possono definire alcuni codici con funzioni speciali, attribuendo loro un nome, a cui si accede con macro del tipo '***&#nome;***'. Spesso si fa uso di queste macro nel DTD, precisamente nelle mappe di sostituzione. Si dovrebbe ricordare che la macro '**&RE;**' fa riferimento convenzionalmente alla fine del record. Si osservi l'esempio seguente:

```
FUNCTION
    RE    13
    RS    10
    SPACE 32
    TAB   SEPCHAR 9
```

Si tratta della direttiva '**FUNCTION**' a cui segue la dichiarazione di una serie di nomi, abbinati al codice relativo. Si può osservare il caso del nome '**TAB**', a cui si aggiunge la parola chiave '**SEPCHAR**': si tratta effettivamente del carattere *<HT>*, che però, ai fini della sintassi concreta, viene tradotto con ciò che corrisponde a '**&SPACE;**', cioè uno spazio normale.

L'esempio mostra la definizione tipica di questa direttiva. Si può osservare che '**RE**' è abbinato a *<CR>*, per cui si suppone che il file sorgente SGML sia organizzato in modo da avere dei codici di interruzione di riga pari a *<CR><LF>*, come avviene nei sistemi Dos. Dipende molto dall'analizzatore SGML come funziona la cosa. In pratica, l'analizzatore potrebbe convertire autonomamente il file in questo modo, oppure potrebbe fare altre considerazioni.

51.3.4.4 Nomi

Attraverso la sottosezione '**NAMING**' è possibile definire quali caratteri possono essere usati nei «nomi». In questo modo si intendono i nomi degli elementi, delle entità, degli attributi e di alcuni tipi di valori da associare agli attributi.

In generale, si fa riferimento alle lettere latine dell'alfabeto inglese e alle cifre numeriche, tenendo conto che in generale è concesso solo di iniziare con una lettera. Per modificare questo assunto si interviene in direttive particolari, che limitano il primo carattere, oppure quelli restanti.

- `LCNMSTRT "`***caratteri_ulteriori***`"`

  `UCNMSTRT "`***caratteri_ulteriori***`"`

  Lower case name start, Upper case name start

  Descrivono rispettivamente il primo carattere minuscolo e maiuscolo. In generale, si indica semplicemente la stringa nulla, '**""**'.

- `LCNMCHAR "`***caratteri_ulteriori***`"`

  `UCNMCHAR "`***caratteri_ulteriori***`"`

  Lower case name characters, Upper case name characters

  Descrivono rispettivamente i caratteri successivi al primo, minuscoli e maiuscoli. In generale, si indica semplicemente la stringa nulla, '**""**'.

- `NAMESTRT `***elenco_codici***

  `NAMECHAR `***elenco_codici***

  Name start, Name characters

  Descrivono rispettivamente i codici utilizzabili nel primo carattere e in quelli restanti. Si usano queste direttive particolarmente nella definizione di XML.

- `NAMECASE`

  Si tratta di un'ulteriore sotto-sottosezione, con la quale si definisce la trasformazione o meno in maiuscolo:

  – `GENERAL YES|NO`

    in questo caso si controlla la conversione in maiuscolo di tutti i nomi, tranne le entità (nell'SGML tradizionale si attiva questa opzione);

  – `ENTITY YES|NO`

    si controlla la conversione in maiuscolo dei nomi di entità e dei loro riferimenti: le macro (nell'SGML tradizionale non si attiva questa opzione).

Nell'SGML normale si utilizza abitualmente la sezione '**NAMING**' nel modo seguente:

```
NAMING
    LCNMSTRT  ""
    UCNMSTRT  ""
    LCNMCHAR  "-."
    UCNMCHAR  "-."
    NAMECASE
        GENERAL     YES
        ENTITY      NO
```

In questo modo, si può osservare che i nomi possono contenere anche il trattino ('**-**') e il punto ('**.**'), ma non possono iniziare così; inoltre, tutti i nomi, tranne quelli delle entità, vengono convertiti in maiuscolo (si parla di ***normalizzazione***), per cui non fa differenza in che modo sono stati scritti.

51.3.4.5 Delimitatori

La sottosezione introdotta dalla parola chiave '**DELIM**' può servire per intervenire nella definizione dei delimitatori. In generale non si modifica nulla e ci si limita a confermare lo standard di riferimento, attraverso la parola chiave '**SGMLREF**':

```
DELIM
    GENERAL   SGMLREF
    SHORTREF  SGMLREF
```

A volte viene disabilitato l'uso delle mappe di sostituzione nel DTD, attraverso la direttiva '**SHORTREF NONE**', come avviene in XML.

Nell'HTML 4 e in XML è stata aggiunta la possibilità di indicare delle macro carattere nella forma '**&x*n*;**', per rappresentare i caratteri attraverso cifre esadecimali. Per ottenere questo risultato, dopo la direttiva '**GENERAL SGMLREF**', si aggiunge la dichiarazione di '**HCRO**':

```
        DELIM
            GENERAL   SGMLREF
            HCRO      "&#38;#x"
            SHORTREF  SGMLREF
```

Naturalmente, in XML ci sono poi altre aggiunte, che qui non vengono mostrate.

51.3.4.6 Nomi riservati

Alcune nomi che hanno significati speciali possono essere modificati nella sottosezione '**NAMES**'. In generale, queste cose non si fanno, per cui si abbina semplicemente la dichiarazione predefinita: '**SGMLREF**':

```
        NAMES
            SGMLREF
```

51.3.4.7 Quantità

Nell'ambito della sintassi concreta è possibile definire il limite a una serie di quantità. Di solito non ci si preoccupa di queste cose, oppure si scrivono direttive per richiedere limiti molto elevati. Per fare riferimento allo standard, si utilizza la parola chiave '**SGMLREF**' come al solito:

```
        QUANTITY SGMLREF
```

Eventualmente si aggiungono le varianti che si ritiene necessario apportare. L'esempio seguente è tratto dalla configurazione predefinita di SP e appare evidente l'intenzione di estendere al massimo i limiti, anche senza spiegare nel dettaglio il significato di ogni parametro:

```
        QUANTITY SGMLREF
            ATTCNT    99999999
            ATTSPLEN  99999999
            DTEMPLEN  24000
            ENTLVL    99999999
            GRPCNT    99999999
            GRPGTCNT  99999999
            GRPLVL    99999999
            LITLEN    24000
            NAMELEN   99999999
            PILEN     24000
            TAGLEN    99999999
            TAGLVL    99999999
```

Con XML, o comunque con la dichiarazione «Web SGML», '**ISO 8879:1986 (WWW)**', è possibile usare una forma differente e più intuitiva per indicare che non si vogliono porre limiti:

```
         QUANTITY NONE
```

51.3.5 Proprietà

L'ultima sezione della dichiarazione SGML serve a raccogliere la definizione delle proprietà: '**FEATURES**'. Contiene in particolare tre sottosezioni intitolate rispettivamente '**MINIMIZE**', '**LINK**' e '**OTHER**'. Non è il caso di approfondire queste definizioni, a parte qualche direttiva che può essere interessante.

Per cominciare, conviene osservare la sezione '**FEATURES**' dell'HTML 4:

```
FEATURES
    MINIMIZE
        DATATAG  NO
        OMITTAG  YES
        RANK     NO
        SHORTTAG YES
    LINK
        SIMPLE   NO
        IMPLICIT NO
        EXPLICIT NO
    OTHER
        CONCUR   NO
        SUBDOC   NO
        FORMAL   YES
```


Nella sottosezione '**MINIMIZE**' è importante tenere in considerazione l'opzione '**DATATAG**', che in generale è bene sia disattivata come appare nell'esempio. Questa dovrebbe servire per specificare una stringa che nel testo deve essere presa in considerazione come una chiusura implicita di un elemento. L'opzione '**OMITTAG**' consente di utilizzare le regole di minimizzazione nel DTD.

La sottosezione '**OTHER**' permette di definire delle caratteristiche interessanti riguardo all'organizzazione del DTD, del sorgente e dei cataloghi. L'opzione '**CONCUR**' consente, se attivata, di gestire più DTD nello stesso documento. Ciò può servire quando è consentita l'aggregazione di più sorgenti che a loro volta utilizzano DTD differenti. Data la complessità che si creerebbe in questo modo, tale opzione viene disabilitata normalmente. L'opzione '**SUBDOC**' permette, se abilitata, di aggregare più sorgenti SGML assieme (che di solito condividono lo stesso DTD implicitamente); se si abilita l'opzione occorre aggiungere l'indicazione del numero massimo di livelli di annidamento a cui si può arrivare. L'opzione '**FORMAL**', se attivata, serve a richiedere l'uso corretto degli identificatori pubblici; se non è attivata, l'identificazione può avvenire in modo meno rigoroso.

L'esempio seguente mostra l'impostazione tradizionale di un sistema SGML:

```
    FEATURES

        MINIMIZE
            DATATAG   NO
            OMITTAG   YES
            RANK      YES
            SHORTTAG  YES

        LINK
            SIMPLE    YES 1000
            IMPLICIT  YES
            EXPLICIT  YES 1

        OTHER
            CONCUR    NO
            SUBDOC    YES 99999999
            FORMAL    YES
```

51.3.6 Applicazione di una dichiarazione SGML in pratica

La dichiarazione SGML può essere attribuita attraverso il catalogo, con la direttiva '**SGMLDECL**':

```
SGMLDECL "HTML4.dcl"
```

L'esempio mostra il riferimento al file '`HTML4.dcl`', contenente la dichiarazione SGML desiderata.

Potrebbe essere impossibile selezionare tra più dichiarazioni alternative. In tal caso, diventa necessario predisporre più cataloghi, uno per ogni tipo di dichiarazione che si intende utilizzare.

51.3.7 Esempio conclusivo

Per concludere viene mostrato un esempio completo di una dichiarazione SGML realizzata per poter utilizzare nel sorgente la codifica ISO 8859-1, che potrebbe essere adatta alle situazioni più comuni (appare anche la sezione '**APPINFO**' che non è stata descritta). Altri esempi possono essere ottenuti dal pacchetto SP sorgente, nel quale si può trovare anche la dichiarazione di XML.

```
<!SGML "ISO 8879:1986 (WWW)"
    CHARSET
        BASESET  "ISO 646-1983//CHARSET
                  International Reference Version
                  (IRV)//ESC 2/5 4/0"
        DESCSET
            0  9 UNUSED
            9  2  9
            11  2 UNUSED
            13  1 13
            14 18 UNUSED
            32 95 32
            127  1 UNUSED
        BASESET  "ISO Registration Number 100//CHARSET
```

```
                ECMA-94 Right Part of
                Latin Alphabet Nr. 1//ESC 2/13 4/1"
        DESCSET  128  32   UNUSED
                 160  96    32

    CAPACITY PUBLIC  "ISO 8879:1986//CAPACITY Reference//EN"

    SCOPE    DOCUMENT

    SYNTAX

        SHUNCHAR CONTROLS 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14
                 15 16 17 18 19 20 21 22 23 24 25 26 27 28
                 29 30 31 127

        BASESET  "ISO 646-1983//CHARSET International
                 Reference Version
                 (IRV)//ESC 2/5 4/0"

        DESCSET
            0 128 0

        FUNCTION
            RE                  13
            RS                  10
            SPACE               32
            TAB       SEPCHAR   9

        NAMING
            LCNMSTRT  ""
            UCNMSTRT  ""
            LCNMCHAR  "-."
            UCNMCHAR  "-."
            NAMECASE
                GENERAL     YES
                ENTITY      NO

        DELIM
            GENERAL   SGMLREF
            HCRO      "&#38;#x"
            SHORTREF  SGMLREF

        NAMES    SGMLREF

        QUANTITY NONE

    FEATURES

        MINIMIZE
            DATATAG   NO
            OMITTAG   YES
            RANK      NO
            SHORTTAG  NO

        LINK
            SIMPLE    YES 1000
            IMPLICIT  YES
            EXPLICIT  YES 1

        OTHER
            CONCUR    NO
            SUBDOC    YES 99999999
            FORMAL    YES

    APPINFO NONE
>
```

51.4 Riferimenti

«

- C. M. Sperberg-McQueen, Lou Burnard, *Guidelines for Electronic Text Encoding and Interchange (TEI P4)*, in particolare il secondo capitolo: *A gentle introduction to XML*, http://www.tei-c.org/release/doc/tei-p4-doc/html/
- *The SGML Newsletter*, http://wayback.archive.org/web/2009*/http://architag.com/tag/
- Oasis, *Core Standards for Markup Language Technologies*, http://xml.coverpages.org/coreStandards.html

- Oasis, *Standard Generalized Markup Language (SGML)*, http://xml.coverpages.org/sgml.html
- Vidar Bronken Gundersen, Rune Mathisen, *SGML/XML character entity reference*, http://www.bitjungle.com/isoent/

[1] In questo momento può apparire strano l'uso di questa forma di eccezione. Tuttavia, per comprenderne meglio il senso, occorrerebbe conoscere come funzionano le entità parametriche che sono descritte più avanti. Con queste si può definire un modello del contenuto attraverso una sorta di variabile e, in tal caso, potrebbe essere conveniente l'indicazione di una o più eccezioni, sia in aggiunta che in detrazione.

XML

XML è un linguaggio derivato dall'SGML, da intendersi come un sottoinsieme «compatibile» con questo; in particolare, il nome rappresenta l'acronimo di *Extensible markup language*. Il motivo per il quale è stata introdotta questa variante dell'SGML è dovuto all'esigenza di trovare un compromesso tra l'SGML originale e l'HTML, che è solo un'applicazione di SGML troppo limitata per la documentazione multimediale. In pratica, l'intento è stato ed è quello di semplificare leggermente l'SGML rendendo disponibili molte qualità dell'SGML che un'applicazione rigida come l'HTML non è in grado di offrire.

In generale, un documento XML è un'***applicazione*** di XML; nello stesso modo, l'HTML (come linguaggio) è un'applicazione SGML.

52.1 Transizione da SGML a XML

XML nasce come sottoinsieme di SGML, ma nel tempo si evolve e si allontana da SGML. In particolare, la prima caratteristica che comincia il distacco tra i due è costituita dai domini applicativi (*namespace*), i quali rendono difficile l'abbinamento di un DTD a un documento XML che ne fa uso.

52.1.1 Codifica

La novità più importante di XML è l'utilizzo predefinito dell'insieme di caratteri universale, prevalentemente attraverso la forma UTF-8 e UTF-16. Questo fatto ha delle implicazioni importanti, in quanto i riferimenti a macro del tipo '`&`***`n`***`;`' e '`&x`***`n`***`;`' si fanno ai punti di codifica dello standard ISO 10646 (nel primo caso il numero è espresso in decimale, mentre nel secondo si tratta di un numero esadecimale).

XML non esclude a priori l'utilizzo di altri tipi di codifica; tuttavia, se non è possibile usare le codifiche UTF-***n***, per evitare ambiguità potrebbe essere conveniente limitarsi all'uso dell'ASCII tradizionale, dal momento che è perfettamente compatibile con la forma UTF-8. Eventualmente è possibile anche specificare il tipo di codifica attraverso un'istruzione apposita, che viene mostrata in seguito.

52.1.2 Commenti

I commenti si indicano in linea di massima come in SGML, attraverso la forma:

```
<-- commento -->
```

Come nell'SGML si deve evitare l'uso di due trattini in sequenza, '`--`', ma in XML non è ammissibile il commento nullo nella forma '`<!>`'.

52.1.3 Marcatori ed elementi vuoti

In XML, gli elementi devono essere aperti e chiusi correttamente attraverso i marcatori relativi; in pratica non è possibile più lasciare all'analizzatore XML il compito di determinare da solo la cosa in base al contesto. Questa limitazione è importante per facilitare il compito dei programmi che devono interpretare un documento XML e comunque si riflette positivamente nella struttura del sorgente del documento stesso.

Gli elementi vuoti vanno indicati regolarmente con il marcatore di chiusura, oppure con un solo marcatore speciale, che ha la forma seguente:

```
<nome_elemento/>
```

In pratica, alla fine del marcatore appare una barra obliqua prima del simbolo '`>`'.

Di fatto, per problemi di compatibilità, si lascia uno spazio prima della barra finale. Per esempio: '`<hr />`'.

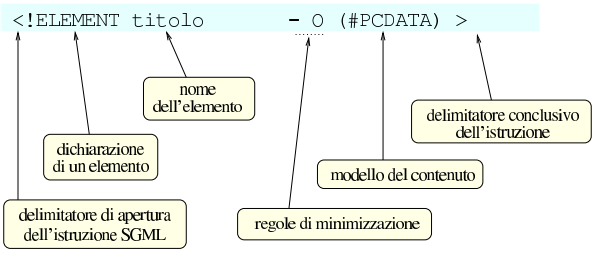
L'assenza della possibilità di definire dei marcatori di apertura o di chiusura opzionali, fa sì che si semplifichi la dichiarazione di questi nel DTD:

```
<ELEMENT nome_elemento modello_del_contenuto >
```

Nella figura 52.1 si vede un confronto tra la dichiarazione SGML e quella XML. Si vede chiaramente che in XML mancano le regole di minimizzazione.

Figura 52.1. Scomposizione delle varie parti della dichiarazione di un elemento SGML e XML.

SGML:



XML:

```
<!ELEMENT titolo (#PCDATA) >
```

In XML, i nomi che si attribuiscono agli elementi e agli attributi sono sensibili alla differenza tra lettere maiuscole e minuscole; per esempio, l'elemento '**testo**' è diverso dall'elemento '**Testo**' e da tutte le altre varianti possibili. Per la precisione, i nomi devono sottostare alle regole seguenti:

- devono iniziare con una lettera alfabetica, oppure con un trattino basso ('_', ovvero #x5F), e possono contenere anche cifre numeriche, il punto, il trattino basso e il trattino normale (#x2D);
- non possono contenere spazi;
- possono contenere i due punti ('**:**'), ma questa possibilità è comunque riservata a situazioni particolari (serve a definire un prefisso che rappresenta un contesto);
- non possono iniziare con la sigla '**xml**', o con qualunque altra variazione delle lettere minuscole e maiuscole, dal momento che questi potrebbero avere in seguito dei significati speciali.

52.1.4 Entità predefinite

Alcune entità standard essenziali sono predefinite e teoricamente non è necessario specificarle nel DTD. Si tratta di '**amp**', '**lt**', '**gt**', '**apos**' e '**quot**'. Le macro relative sono '**&**', '**<**', '**>**', '**'**' e '**"**'.

Si può osservare questo particolare nella dichiarazione SGML di XML:

```
SYNTAX
    ...
    ENTITIES
        "amp"  38
        "lt"   60
        "gt"   62
        "quot" 34
        "apos" 39
```

52.1.5 Entità parametriche

In XML, le entità parametriche possono essere utilizzate solo all'interno del DTD. Da ciò consegue logicamente che le sezioni marcate con le quali si può includere o escludere del testo in base al contenuto di un'entità parametrica, possono esistere solo nel DTD.


```
<!ENTITY % bozza 'INCLUDE' >
<!ENTITY % finale 'IGNORE' >

<![%bozza;[
<!ELEMENT libro (commento*, titolo, corpo)>
]]>
<![%finale;[
<!ELEMENT libro (titolo, corpo)>
]]>
```

L'esempio mostra un pezzo di un DTD ipotetico, in cui vengono dichiarate due entità parametriche, '**bozza**' e '**finale**'. In questo caso, la macro '**%bozza;**' si traduce nella parola '**INCLUDE**', mentre la macro '**%finale;**' si traduce nella parola '**IGNORE**'. In questo modo, viene dichiarato l'elemento '**libro**' nella prima modalità: quella che ammette la presenza dell'elemento '**commento**'.

52.1.6 Altre sezioni marcate

XML ammette l'uso di un'altra sezione marcata soltanto, la sezione '**CDATA**' per delimitare del testo letterale.

```
<![CDATA[Il marcatore <ciao> serve per...]]>
```

L'esempio mostra in che modo sia possibile utilizzare letteralmente i simboli '**<**' e '**>**' in una sezione '**CDATA**'.

52.1.7 Istruzioni di elaborazione

Le istruzioni di elaborazione sono una novità in XML. Servono in qualche modo per passare delle informazioni alle applicazioni. Si distinguono per avere la forma seguente:

```
<?istruzione_di_elaborazione?>
```

Il testo che compone l'istruzione dipende dall'applicazione a cui è diretto. È importante tenere presente che tutto ciò che inizia con la stringa '**xml**', assieme a tutte le sue variazioni di lettere maiuscole e minuscole, è riservato.

In generale, in base al significato che può avere l'istruzione di elaborazione, queste possono trovarsi in qualunque parte del sorgente XML.

Normalmente si inizia sempre un sorgente XML con un'istruzione di elaborazione che dichiara la versione di XML a cui si fa riferimento, assieme alla codifica utilizzata:

```
<?xml version="1.0" encoding="UTF-8"?>
```

52.1.8 Convenzioni dell'XML

Nella descrizione fatta fino a questo punto sono già state presentate alcune convenzioni di XML che non sono esprimibili nella dichiarazione SGML relativa. In pratica, si tratta di regole che vanno tenute in considerazione quando si scrive un DTD per un documento XML. Vale la pena di raccogliere le convenzioni più importanti.

- I nomi di elementi e degli attributi che iniziano per '**xml**', con qualsiasi altra variante delle lettere maiuscole e minuscole, sono riservati.
- Gli elementi che ne possono avere bisogno, devono poter disporre di un attributo denominato '**xml:space**', a cui possano essere assegnate le parole chiave '**default**' o '**preserve**'. Il suo scopo è quello di definire il comportamento nei confronti degli spazi (di tutti i caratteri assimilabili a questo concetto). Assegnando la parola chiave '**default**' si intende lasciare che gli spazi vengano gestiti come al solito, eliminando quelli superflui; con la parola chiave '**preserve**' si vuole richiedere di mantenere gli spazi come sono. La dichiarazione di questo attributo può avvenire nel DTD come nell'esempio seguente:

  ```
  <!ATTLIST esempio xml:space (default|preserve) 'preserve'>
  ```

  In particolare, un elemento che per sua natura deve rispettare le spaziature originali, potrebbe essere definito nel modo seguente, dove si vede il caso dell'elemento '**pre**' di XHTML:

```
<!ELEMENT pre %pre.content;>
<!ATTLIST pre
  %attrs;
  xml:space (preserve) #FIXED 'preserve'
>
```

- Gli elementi che ne possono avere bisogno, devono poter disporre di un attributo denominato '**`xml:lang`**', a cui poter assegnare un codice identificativo del linguaggio contenuto. Si prevede l'uso di diversi tipi di codice:
  - un codice di linguaggio composto da due lettere, secondo lo standard ISO 639 (tabella 13.4);
  - un codice di linguaggio registrato dall'autorità IANA (*Internet assigned numbers authority*), a cui va aggiunto comunque il prefisso '**`i-`**', oppure '**`I-`**';
  - un codice stabilito dall'utente o concordato tra le parti, a cui va aggiunto il prefisso '**`x-`**', oppure '**`X-`**'.

  La dichiarazione di questo attributo può avvenire nel DTD come nell'esempio seguente:

```
<!ATTLIST esempio xml:lang NMTOKEN #IMPLIED >
```

  Eventualmente si può anche specificare un linguaggio predefinito, come si vede nell'esempio seguente:

```
<!ATTLIST testo xml:lang NMTOKEN 'it' >
```

52.1.9 Correttezza formale e validità

Possono esistere due livelli di approccio all'XML da parte dei programmi che lo utilizzano: il primo si limita a leggere il documento senza sapere nulla della sua struttura stabilita nel DTD; il secondo invece richiede la conoscenza di questa struttura. Nel primo caso è sufficiente che il documento XML sia stato scritto correttamente dal punto di vista formale, in senso generale; in questo modo si parla di *well formed document*. Nel secondo caso è importante che il documento, oltre che essere corretto dal punto di vista formale, sia anche valido in base alla definizione stabilita nel DTD.

Il documento XML corretto dal punto di vista formale, ha le caratteristiche seguenti:

- contiene un elemento principale unico, all'interno del quale vanno collocati tutti gli altri (si parla comunemente dell'elemento *root*, ovvero della radice);
- tutti i marcatori degli elementi devono essere indicati in modo corretto, attraverso degli annidamenti ordinati;
- tutti gli elementi devono essere delimitati correttamente, senza saltare dei marcatori, inoltre gli elementi vuoti vanno chiusi oppure vanno indicati con il marcatore speciale già mostrato;
- devono essere rispettate le regole stabilite per i nomi degli elementi;
- i valori associati agli attributi vanno delimitati sempre attraverso apici doppi oppure apici singoli;

Il documento XML valido, oltre a essere corretto formalmente, deve anche essere conforme al DTD. Come nell'SGML normale, il DTD può essere indicato attraverso un riferimento, oppure può essere incorporato all'inizio del documento.

52.1.10 Verifica della validità con SP

Il pacchetto SP di James Clark può essere utilizzato anche per convalidare un documento XML, a partire dal suo DTD (il procedimento è analogo a quanto già mostrato nella sezione 51.2); tuttavia, è necessario procurarsi la dichiarazione XML, che si può trovare nell'archivio dei sorgenti di SP stesso: '`pubtext/xml.dcl`'.

Supponendo di disporre del file '`xml.dcl`' nella directory corrente, si può realizzare un catalogo molto semplice come quello seguente:

```
SGMLDECL "xml.dcl"
```


Naturalmente, nel catalogo si possono aggiungere anche altre cose, in base alla necessità o meno di indicare il DTD e le entità generali. Per verificare il funzionamento della cosa, si può provare a eseguire la convalida dell'esempio seguente, che include il DTD nel preambolo e non ha bisogno di entità generali:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE esempio [
    <!ELEMENT esempio (#PCDATA)>
]>
<esempio>Ciao a tutti!</esempio>
```

Si può osservare che si tratta di un documento elementare, in cui esiste solo l'elemento principale, denominato '**`esempio`**'.

Per la convalida, si può usare l'eseguibile '**`nsgmls`**' nel modo seguente:

```
$ nsgmls -c catalogo.xml -s esempio.xml [Invio]
```

Qui si sottintende che il file del catalogo sia '`catalogo.xml`' e che il sorgente XML sia contenuto nel file '`esempio.xml`'. Se oltre alla convalida si vuole avere il risultato pre-elaborato, si toglie l'opzione '**`-s`**', ottenendo quanto segue:

```
?xml version="1.0" encoding="ISO-8859-1"
(esempio
-Ciao a tutti!
)esempio
C
```

52.1.11 Domini applicativi: «namespace»

Con il termine ***dominio applicativo*** si vuole qui fare riferimento a quello che è noto come *namespace* a proposito di XML. Si definisce un dominio applicativo associando elementi, ma eventualmente anche attributi, a qualcosa che viene identificato tramite un indirizzo URI (sia URL, sia URN; si veda anche la sezione 54.1 a proposito dell'estensione del termine URI). Questa associazione ha lo scopo di evitare ambiguità, quando per qualche ragione si utilizzerebbero elementi o attributi con lo stesso nome, ma con significati differenti.

Dal momento che non è possibile utilizzare nomi di elementi e di attributi che contengano direttamente un URI, si associa questo URI attraverso un attributo particolare:

```
1  <?xml version="1.0"?>
2  <html xmlns="http://www.w3.org/1999/xhtml">
3    <head>...</head>
4    <body>
5      <h1>Bla bla bla</h1>
6      ...
7      <math xmlns="http://www.w3.org/1998/Math/MathML">
8        <mi>x</mi><mo>+</mo><mn>3</mn>
9      </math>
10   </body>
11 </html>
```

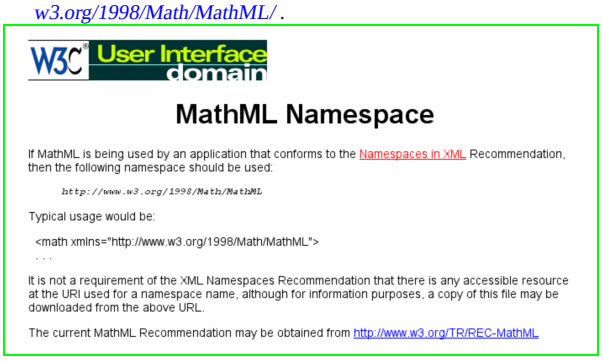
Questo esempio mostra l'utilizzo di MathML all'interno di un documento XHTML (le righe dalla settima alla nona). Si può osservare che l'elemento '**`html`**', alla riga numero due dell'esempio, contiene l'attributo '**`xmlns`**' a cui è associata la stringa '**`http://www.w3.org/1999/xhtml`**'. Questa dichiarazione specifica che all'elemento '**`html`**' e a tutti i suoi discendenti si associa il dominio applicativo *http://www.w3.org/1999/xhtml* . Questo indirizzo (*http://www.w3.org/1999/xhtml*) è solo un punto di riferimento univoco; se si vuole, lo si può anche visitare (figura 52.14), ma ha soltanto lo scopo di dichiarare che gli elementi a cui è associato si riferiscono a XHTML; inoltre, non è nemmeno necessario che esista veramente.

Con i linguaggi SGML, si usa normalmente la dichiarazione del DTD a cui si fa riferimento. Anche con XML, volendo, è corretto usare un riferimento al DTD con l'istruzione '**`<DOCTYPE…>`**', ma si inserisce questo concetto nuovo del dominio applicativo.

Figura 52.14. Come si presenta alla vista l'indirizzo *http://www.w3.org/1999/xhtml* .

Continuando a leggere l'esempio mostrato, alla settima riga appare l'elemento '**`math`**', che, come dichiarato dall'attributo '**`xmlns`**' appartiene al dominio applicativo *http://www.w3.org/1998/Math/MathML/* ; in pratica, dichiara che l'elemento in questione e ciò che contiene riguarda MathML.

Figura 52.15. Come si presenta alla vista l'indirizzo *http://www.w3.org/1998/Math/MathML/* .



Quanto visto in questo esempio, rappresenta un uso «predefinito» del dominio applicativo. Infatti, esiste anche la possibilità di associare gli elementi (e gli attributi) a un dominio applicativo, specificando un prefisso:

```
1  <xsl:stylesheet
2       xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
3    <xsl:template name="/">
4      ...
5    </xsl:template>
6  </xsl:stylesheet>
```

In questo caso si può osservare, alla riga numero uno, che l'elemento '**`xsl:stylesheet`**' contiene l'attributo '**`xmlns:xsl`**', a cui è associato un URI. Ciò significa che, nell'ambito del controllo dell'elemento '**`xsl:stylesheet`**', elementi e attributi il cui nome inizia con il prefisso '**`xsl:`**' appartengono al dominio applicativo *http://www.w3.org/1999/XSL/Transform* ; per la precisione, anche lo stesso elemento '**`xsl:stylesheet`**' appartiene a questo dominio applicativo, proprio perché anche il suo nome inizia con il prefisso '**`xsl:`**'.


```
1  <?xml version="1.0" encoding="utf-8"?>
2  <xsl:stylesheet version="1.0"
3          xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
4          xmlns:fo="http://www.w3.org/1999/XSL/Format">
5
6  <xsl:output method="xml" indent="yes"/>
7  ...
8  <xsl:template match="studio">
9  <fo:root>
10     <fo:layout-master-set>
11         ...
12     </fo:layout-master-set>
13     <fo:page-sequence master-reference="cover">
14         ...
15     </fo:page-sequence>
16     ...
17 </fo:root>
18 </xsl:template>
19 ...
20 </xsl:stylesheet>
```

Questo nuovo esempio mostra l'uso di due domini applicativi, associati rispettivamente ai prefissi '**`xsl:`**' e '**`fo:`**' (le righe tre e quattro). La dichiarazione dei domini applicativi avviene nell'elemento principale, ovvero '**`xsl:stylesheet`**' (che appartiene al dominio applicativo associato al prefisso '**`xsl:`**'), in modo che possa essere valida per tutto il documento, salva la possibilità di modificarla all'interno di elementi ben precisi.

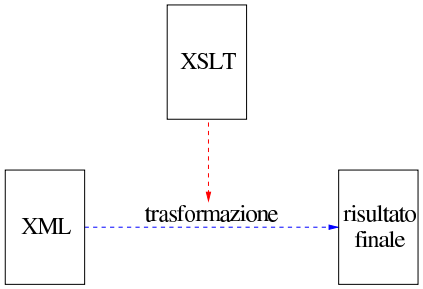
Oltre ai modi mostrati in questi esempi, esistono altre combinazioni; per esempio è possibile affiancare un dominio applicativo predefinito a domini specificati attraverso l'uso di prefissi. Tuttavia si preferisce evitare di creare troppa confusione su un concetto che invece dovrebbe essere semplice, per consentire la comprensione di altri concetti legati a XML. Eventualmente si possono consultare i documenti annotati nella bibliografia al termine del capitolo, per una visione più precisa e dettagliata a proposito dei domini applicativi (ovvero *namespace*).

La presenza dei domini applicativi rende difficile l'abbinamento di un DTD a un documento XML, perché i prefissi devono essere previsti esattamente nel DTD, come parte dei nomi degli elementi e degli attributi, mentre la definizione di un dominio applicativo consentirebbe l'utilizzo di un prefisso libero, deciso nel momento in cui si usa l'attributo '**`xmlns:x`**'.

52.2 Introduzione ai fogli di stile XSLT

XSLT è un linguaggio realizzato in forma di file XML, con il quale si definisce una trasformazione di un file XML in un altro file. Generalmente, il file di destinazione è un altro file XML, anche se può comunque essere qualcosa di diverso. La sigla sta precisamente per *XSL transformations* (ovvero *Extensible stylesheet language transformations*), a indicare che il linguaggio è scritto in quello che viene chiamato «foglio di stile XSL» e serve a trasformare i dati originali in qualcosa di conveniente ai propri scopi. In altri termini, si tratta di un linguaggio che consente di estrarre le informazioni contenute in un file XML, per generare con queste ciò che serve.

Figura 52.18. Utilizzo del linguaggio XSLT per la trasformazione di un file XML.



Naturalmente, l'elaborazione di un file XML secondo il linguaggio XSLT richiede un programma apposito. Qui viene mostrato l'uso di

Xalan (sezione 52.2.6).

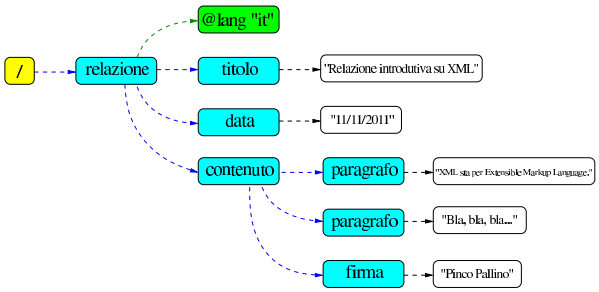
La trattazione che qui viene fatta a proposito dei fogli di stile XSLT è limitata alle funzionalità principali. Per un approfondimento si può consultare la documentazione elencata nella bibliografia alla fine del capitolo.

52.2.1 Struttura ad albero, percorsi e modelli di confronto

Un documento XML, ai fini della sua elaborazione, viene visto come una struttura ad albero, dove ogni componente rappresenta un nodo di questo. Si osservi l'esempio seguente, che rappresenta un file XML molto semplice:

```
<?xml version="1.0" encoding="UTF-8" ?>
<relazione lang="it">
<titolo>Relazione introduttiva su XML</titolo>
<data>11/11/2011</data>
<contenuto>
<paragrafo>XML sta per Extensible Markup Language.</paragrafo>
<paragrafo>Bla, bla, bla...</paragrafo>
<firma>Pinco Pallino</firma>
</contenuto>
</relazione>
```

Si potrebbe rappresentare schematicamente l'albero di questo documento come si vede nella figura successiva:



Nello schema mostrato si deve osservare che elementi, attributi e testo contenuto negli elementi, costituiscono nodi separati.

Per identificare un nodo dell'albero, si usa una notazione che assomiglia ai percorsi dei file system Unix. A titolo di esempio vengono mostrati tutti gli elementi e gli attributi XML degli schemi già apparsi, secondo la sequenza originale, con percorsi assoluti:

```
/
/relazione
/relazione/@lang
/relazione/titolo
/relazione/data
/relazione/contenuto
/relazione/contenuto/paragrafo
/relazione/contenuto/paragrafo
/relazione/contenuto/firma
```

Naturalmente esistono anche percorsi relativi, quando manca la barra obliqua iniziale che rappresenta la radice: questi fanno riferimento al nodo corrente nell'ambito del contesto a cui ci si riferisce.

La sintassi con cui si possono definire questi percorsi è stabilita dal linguaggio XPath, ovvero *XML path language*. Si tratta di un sistema abbastanza complesso che non viene mostrato qui nel dettaglio. Per quanto riguarda il linguaggio XSLT i percorsi vengono usati per definire un modello di confronto con i nodi di un documento XML; nell'ambito di questi modelli di confronto si utilizzano delle notazioni particolari rispetto alla convenzione generale costituita da XPath. La tabella 52.22 riporta alcuni esempi di questi modelli.

Tabella 52.22. Alcuni modelli di confronto per individuare i nodi di un documento XML.

Modello	Descrizione
`/`	Una barra obliqua da sola individua il nodo radice.
`text()`	Individua qualunque nodo di testo.
`node()`	Individua qualunque nodo, esclusi gli attributi e la radice.
`.` `self::node()`	Una di queste due notazioni, indifferentemente, individua il nodo corrente.
`..` `parent::node()`	Una di queste due notazioni, indifferentemente, individua il nodo genitore di quello corrente.
elemento `child::`***elemento***	Un nome scritto senza l'inserzione di simboli speciali rappresenta un elemento. La seconda notazione è completa.
***elemento*`[`*n*`]`	Individua l'***n***-esimo elemento con quel nome, che si trova all'interno dell'elemento che lo contiene. Osservando gli esempi mostrati in precedenza, '**`paragrafo[1]`**' può rappresentare il primo elemento '**`paragrafo`**' che si trova all'interno dell'elemento '**`contenuto`**'.
@attributo `attribute::`***attributo***	Un nome preceduto da una chiocciolina rappresenta un attributo. La seconda notazione è completa.
`*`	L'asterisco corrisponde a qualunque elemento.
`@*`	L'asterisco preceduto da una chiocciolina corrisponde a qualunque attributo.
***elemento_1*`	`*elemento_2***	I nomi di due elementi, separati da una barra verticale, indicano la corrispondenza con l'uno o con l'altro.
***@attributo_1*`	`*@attributo_2***	I nomi di due attributi, separati da una barra verticale, indicano la corrispondenza con l'uno o con l'altro.
elemento_1*`/`*elemento_2	La barra obliqua separa i nodi di un percorso; in questo caso, si vuole individuare l'elemento ***elemento_2*** che è contenuto da ***elemento_1***.
elemento*`/`*@attributo	In questo caso si vuole fare riferimento a un attributo di un certo elemento.
elemento_1*`//`*elemento_2 ***elemento_1***`↩` `↪/descendant-or-self::node()/`↩ `↪`***elemento_2***	Due barre oblique indicano una discendenza non meglio precisata: qui, ***elemento_2*** è contenuto all'interno di ***elemento_1***, direttamente oppure all'interno di annidamenti ulteriori.

52.2.2 Struttura generale del foglio di stile XSLT

Un foglio di stile XSLT è un documento XML con una struttura particolare, abbinato al dominio applicativo *http://www.w3.org/1999/XSL/Transform* (sezione 52.1.11). Normalmente, il prefisso associato a questo dominio applicativo è '**`xsl`**', pertanto, l'elemento principale è '**`xsl:stylesheet`**', oppure '**`xsl:transform`**', indifferentemente:

```
<xsl:stylesheet
    version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    [altri_attributi]>
    [contenuto]
</xsl:stylesheet>
```

```
<xsl:transform
    version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    [altri_attributi]>
    [contenuto]
</xsl:transform>
```

Anche senza definire esplicitamente alcun tipo di trasformazione, si ottiene ugualmente un risultato elaborando un file XML, attraverso delle regole di trasformazione predefinite, con le quali, in buona sostanza, si ottiene il testo del file XML, senza i marcatori che delimitano gli elementi. Per esempio, si può prendere il file XML già presentato all'inizio del capitolo, associandogli il foglio di stile seguente:

```
<xsl:stylesheet
    version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
</xsl:stylesheet>
```

Supponendo che il file XML si chiami '`prova.xml`' e che il foglio di stile sia '`prova.xsl`', si potrebbe usare Xalan nel modo seguente:

```
$ xalan -IN prova.xml -XSL prova.xsl -OUT prova.txt [Invio]
```

Ecco cosa si ottiene nel file '`prova.txt`':

```
<?xml version="1.0" encoding="UTF-8"?>

Relazione introduttiva su XML
11/11/2011

XML sta per Extensible Markup Language.
Bla, bla, bla...
Pinco Pallino
```

52.2.3 Modelli di confronto e trasformazione

La definizione di un criterio di trasformazione dei nodi del documento XML di origine in quello che si vuole ottenere, avviene principalmente per mezzo di modelli di confronto, attraverso un elemento del foglio di stile denominato '**`xsl:template`**'. La traduzione del termine *template*, ovvero «mascherina», rende bene l'idea del concetto: gli elementi '**`xsl:template`**' del foglio di stile definiscono un modello di confronto con cui selezionano alcuni nodi del documento XML di origine; su questi nodi applicano delle trasformazioni.

```
<xsl:template
    match="modello_di_confronto"
    [altri_attributi]>
    trasformazione
</xsl:template>
```

Il modello di confronto viene definito secondo le regole che in parte sono state descritte nella tabella 52.22; per esempio, il blocco seguente individua il nodo radice:

```
<xsl:template match="/">
    ...
    ...
</xsl:template>
```

Per comprendere meglio cosa accade, si prenda il solito esempio di file XML già considerato in precedenza e si applichi il foglio di stile seguente:


```
<xsl:stylesheet
    version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
    <xsl:template match="/relazione/contenuto">
        ciao ciao
    </xsl:template>
</xsl:stylesheet>
```

Come si vede, è stato inserito un solo elemento '**`xml:template`**' che seleziona l'elemento '**`contenuto`**', che si trova all'interno di '**`relazione`**' (che a sua volta è l'elemento principale del documento XML). Quando si individua questo elemento, viene inserito il testo «ciao ciao». Valgono le stesse convenzioni dei nomi dei file già viste in precedenza:

```
$ xalan -IN prova.xml -XSL prova.xsl -OUT prova.txt [Invio]
```

Ecco cosa si ottiene nel file '`prova.txt`':

```
<?xml version="1.0" encoding="UTF-8"?>

Relazione introduttiva su XML
11/11/2011

        ciao ciao
```

In pratica, le regole di trasformazione predefinite hanno inserito il contenuto degli elementi '**`titolo`**' e '**`data`**'. Quindi, l'elemento '**`contenuto`**', con tutto quello che esiste al suo interno, è stato sostituito con la stringa «ciao ciao».

Per fare in modo che vengano presi in considerazione anche gli elementi contenuti all'interno di ciò che viene individuato, si utilizza l'elemento '**`xsl:apply-templates`**', che di solito è vuoto:

```
<xsl:template
    match="modello_di_confronto"
    [altri_attributi]>
    …
    <xsl:apply-templates
        [select="modello_di_selezione"]
        [altri_attributi]>
        …
    </xsl:apply-templates>
    …
</xsl:template>
```

Per esempio, se il foglio di stile di prova viene modificato nel modo seguente:

```
<xsl:stylesheet
    version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
    <xsl:template match="/relazione/contenuto">
        prima
        <xsl:apply-templates />
        dopo
    </xsl:template>
</xsl:stylesheet>
```

La trasformazione genera questo file:

```
<?xml version="1.0" encoding="UTF-8"?>

Relazione introduttiva su XML
11/11/2011

        prima

XML sta per Extensible Markup Language.
Bla, bla, bla...
Pinco Pallino

        dopo
```

In pratica, si consente alle regole di trasformazione predefinite (dal momento che non ce ne sono altre nell'esempio) di occuparsi degli elementi contenuti all'interno dell'elemento '**`contenuto`**', cosa che produce semplicemente l'estrazione del testo che questi circoscrivono. Tutto questo avviene perché «l'istruzione» '**`apply-templates`**'

serve proprio a richiamare gli altri modelli di confronto (e trasformazione) per quanto è contenuto in ciò che è stato individuato.

Dal momento che nell'esempio non ci sono altri modelli di confronto e trasformazione, è evidente che si tratta soltanto di quelli predefiniti.

L'elemento '**xsl:apply-templates**' può avere un attributo molto importante, '**select**', che consente di specificare su cosa continuare il confronto con altri modelli; in pratica consente di limitare, o controllare, la ricorsione. Si osservi la variante seguente del foglio di stile:

```
<xsl:stylesheet
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    version="1.0">
    <xsl:template match="/relazione/contenuto">
        prima
        <xsl:apply-templates select="paragrafo" />
        dopo
    </xsl:template>
</xsl:stylesheet>
```

Per prima cosa è bene osservare che il modello indicato con l'attributo '**select**' rappresenta un percorso relativo, che secondo il contesto è riferito esattamente a '`/relazione/contenuto/paragrafo`'. In base a questa selezione, si vuole che nell'ambito del contenuto dell'elemento (o degli elementi) '`/relazione/contenuto`', la ricorsione successiva prenda in considerazione soltanto gli elementi '**paragrafo**'. Ecco cosa si ottiene; come si vede, il testo dell'elemento '**firma**' è stato ignorato:

```
<?xml version="1.0" encoding="UTF-8"?>

Relazione introduttiva su XML
11/11/2011

        prima
        XML sta per Extensible Markup Language.Bla, bla, bla...
        dopo
```

Naturalmente, è possibile utilizzare '**xsl:apply-templates**' più volte all'interno dello stesso modello di confronto e trasformazione, anche così:

```
<xsl:stylesheet
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    version="1.0">
    <xsl:template match="/relazione/contenuto">
        Io sottoscritto, <xsl:apply-templates select="firma" />,
        affermo che:
        <xsl:apply-templates select="paragrafo" />

        Firmato:
        <xsl:apply-templates select="firma" />
    </xsl:template>
</xsl:stylesheet>
```

Ecco il risultato che si può ottenere:

```
<?xml version="1.0" encoding="UTF-8"?>

Relazione introduttiva su XML
11/11/2011

        Io sottoscritto, Pinco Pallino,
        affermo che:
        XML sta per Extensible Markup Language.Bla, bla, bla...

        Firmato:
        Pinco Pallino
```

I modelli di confronto e trasformazione possono basarsi su riferimenti relativi ai nodi, se non ha importanza la collocazione esatta di questi nell'albero del documento XML di origine. Si osservi l'esempio seguente:


```
<xsl:stylesheet
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    version="1.0">
    <xsl:template match="paragrafo">
        "<xsl:apply-templates />"
    </xsl:template>
</xsl:stylesheet>
```

In questo caso, si evidenziano gli elementi '**paragrafo**', attorno ai quali si vuole appaiano gli apici doppi; tutto il resto viene gestito in modo predefinito:

```
<?xml version="1.0" encoding="UTF-8"?>

Relazione introduttiva su XML
11/11/2011


        "XML sta per Extensible Markup Language."


        "Bla, bla, bla..."

Pinco Pallino
```

Come accennato, l'elemento '**xsl:apply-templates**' del foglio di stile XSLT, normalmente è vuoto. Quando contiene qualcosa, ciò serve per elaborare il risultato della scansione che lo riguarda. In pratica, '**xsl:apply-templates**' serve a richiamare i modelli di confronto e trasformazione successivi, in modo da inserire il risultato di queste elaborazioni nel punto in cui si trova, ma se questo elemento non è vuoto, le «istruzioni» che contiene servono a interferire con le elaborazioni successive, per esempio per riordinare i dati ottenuti. Questo tipo di utilizzo di '**xsl:apply-templates**' non viene descritto.

52.2.4 Testo

Negli esempi delle sezioni precedenti, in varie occasioni è stato inserito del testo aggiuntivo nella trasformazione del documento XML di origine. In questo modo è possibile anche aggiungere marcatori e altre componenti, in base agli scopi che ci si prefigge con la trasformazione.

Quando un elemento contiene solo spazi bianchi (spazi veri e propri, tabulazioni orizzontali e interruzioni di riga), il nodo relativo può essere eliminato dalla struttura. In generale, salvo che sia specificato diversamente, nel documento XML di origine vengono preservati tutti gli spazi, mentre nel foglio di stile sono preservati solo quelli contenuti negli elementi '**xsl:text**'.

52.2.4.1 Formato generale del risultato della trasformazione

Esistono diverse modalità di funzionamento a cui si adegua la trasformazione del documento XML, in base al tipo di file che si vuole ottenere. Questa modalità di funzionamento si seleziona con l'elemento '**xsl:output**':

```
<xsl:output
    method="metodo"
    version="versione"
    encoding="codifica"
    omit-xml-declaration="yes"|"no"
    doctype-public="dichiarazione_dtd_public"
    doctype-system="dichiarazione_dtd_system"
    [altri_attributi] />
```

Come si vede dal modello sintattico, si tratta di un elemento del foglio di stile che non ha lo scopo di contenere qualcosa e tutto è indicato attraverso attributi.

L'attributo più importante è '**method**', al quale si associano normalmente le parole chiave '**xml**', '**html**' e '**text**', per indicare rispettivamente che si vuole ottenere un risultato in formato XML, HTML o un file di testo senza una conformazione particolare conosciuta. In

generale, se non si specifica il formato del file che si va a generare, si intende XML, cosa che dovrebbe spiegare il motivo per cui negli esempi mostrati in precedenza appare la dichiarazione XML anche se il file ottenuto è semplicemente un testo puro e semplice.

Come si può intuire, a seconda del tipo di «metodo» prescelto, gli altri attributi possono acquisire o perdere significato.

Tabella 52.36. Alcuni attributi di '**xsl:output**' utili quando il risultato dell'elaborazione deve essere un file XML.

Attributo	Descrizione
`method="xml"`	Dichiara il metodo di trasformazione in un documento XML. Alcuni caratteri (come '**&**' e '**<**') vengono trasformati nelle rispettive entità generali standard.
`version="`***versione***`"`	Dichiara il numero di versione del formato XML. Il valore predefinito è '**1.0**'.
`encoding="`***codifica***`"`	Dichiara il tipo di codifica del file da generare. Sono valide sempre le sigle '**UTF-8**' e '**UTF-16**', ma possono essere disponibili altre sigle, come per esempio '**ISO-8859-*n***'.
`omit-xml-declaration=` `"yes"|"no"`	Consente di omettere (se si assegna il valore '**yes**') la dichiarazione XML iniziale.
`indent="yes"|"no"`	Permette di richiedere un incolonnamento degli elementi nel risultato della trasformazione.
`doctype-public="`***dtd_public***`"`	Permette di specificare la dichiarazione del DTD, utilizzando un identificatore pubblico, come potrebbe essere: '**-//W3C//DTD XHTML 1.1//EN**'.
`doctype-system="`***dtd_system***`"`	Permette di specificare la dichiarazione del DTD, indicando come riferimento un file.

Tabella 52.37. Alcuni attributi di '**xsl:output**' utili quando il risultato dell'elaborazione deve essere un file HTML.

Attributo	Descrizione
`method="html"`	Dichiara il metodo di trasformazione in un documento HTML. Alcuni caratteri vengono trasformati usando le entità generali standard.
`version="`***versione***`"`	Dichiara il numero di versione del formato HTML. Il conoscere la versione permette di verificare la validità del foglio di stile anche in base alla validità del codice HTML che si va a generare. Il valore predefinito è '**4.0**'.
`encoding="`***codifica***`"`	Dichiara il tipo di codifica del file da generare. Sono valide sempre le sigle '**UTF-8**' e '**UTF-16**', ma possono essere disponibili altre sigle, come per esempio '**ISO-8859-*n***'.
`doctype-public="`***dtd_public***`"`	Permette di specificare la dichiarazione del DTD, utilizzando un identificatore pubblico, come potrebbe essere: '**ISO/IEC 15445:2000//DTD HTML//EN**'.
`doctype-system="`***dtd_system***`"`	Permette di specificare la dichiarazione del DTD, indicando come riferimento un file.

Tabella 52.38. Alcuni attributi di '**xsl:output**' utili quando il risultato dell'elaborazione deve essere un file di testo non meglio identificabile.

Attributo	Descrizione
`method="text"`	Dichiara il metodo di trasformazione in un file di testo. In questo caso non avviene alcuna trasformazione dei caratteri, come invece può avvenire con le trasformazioni in XML o in HTML.
`encoding="`***codifica***`"`	Dichiara il tipo di codifica del file da generare. Sono valide sempre le sigle '**UTF-8**' e '**UTF-16**', ma possono essere disponibili altre sigle, come per esempio '**ISO-8859-*n***'.

52.2.4.2 Inserimento di testo aggiuntivo

Generalmente è sufficiente aggiungere del testo estraneo all'interno del foglio di stile XSLT, nell'ambito degli elementi '**xsl:template**', per fare in modo che questo venga inserito nel risultato finale. Per avere un controllo maggiore si può usare l'elemento '**xsl:text**', che tra le altre cose permette di inserire blocchi di spazi quando diversamente verrebbero eliminati:

```
<xsl:text
    [disable-output-escaping="yes"|"no"]>
    testo
</xsl:text>
```

L'attributo '**disable-output-escaping**' permette, se si assegna il valore '**yes**', di disabilitare la sostituzione di alcuni caratteri in entità generali standard (questo vale solo per le trasformazioni che prevedono nella destinazione un formato XML o HTML, perché nel caso di trasformazioni in formato testo, questa sostituzione non viene mai eseguita). Come si può intendere, la funzionalità è disabilitata in modo predefinito, pertanto, normalmente si ottiene la sostituzione di questi caratteri.

In alcune circostanze è necessario inserire del testo nel foglio di stile XSLT che non deve essere interpretato prima della trasformazione, lasciandolo in modo letterale. Per questo si usa normalmente una sezione marcata di tipo CDATA, come nell'esempio seguente, dove si vede l'intenzione di inserire uno stile CSS nel documento finale che è un file HTML:

```
    <xsl:template match="/">
        <HTML>
        <HEAD>
            <TITLE>senza titolo</TITLE>
        <xsl:text disable-output-escaping="yes">
<![CDATA[<STYLE TYPE="text/css">
<!--
BODY {
    background-color: rgb(255, 255, 255)
}
H1 {
    text-align:       right;
    color:            rgb(230, 100, 180)
}
-->
</STYLE>]]>
        </HEAD>
        <BODY>
        <xsl:apply-templates />
        </BODY>
    </xsl:template>
```

52.2.4.3 Estrazione del valore contenuto in un nodo

Il testo contenuto in un nodo viene inserito nel documento finale attraverso l'elemento '**xsl:value-of**' del foglio di stile. Esiste anche un modello di confronto e trasformazione predefinito che esegue questa operazione per tutti i nodi di testo, pertanto, negli esempi di fogli di stile non si è ancora presentata la necessità di mostrarne l'uso:

```
<xsl:value-of
    select="modello_di_selezione"
    [disable-output-escaping="yes"|"no"] />
```

Come si vede dal modello sintattico, l'elemento '**xsl:value-of**' si usa vuoto e, per ottenere il «valore» di qualcosa, occorre specificar-

lo attraverso un modello di selezione, come avviene con l'elemento '**xsl:apply-templates**'.

Anche questo elemento prevede l'attributo '**disable-output-escaping**' come descritto a proposito di '**xsl:text**'.

L'elemento '**xsl:value-of**' si può usare per estrarre il testo contenuto in un elemento del documento di origine, oppure per fare altrettanto da un attributo; in generale, è più frequente l'uso di '**xsl:value-of**' per estrarre il testo di un attributo, come nell'esempio seguente:

```
<xsl:template match="relazione">
    Questo documento utilizza il linguaggio <xsl:value-of
    select="@lang" />.
</xsl:template>
```

In questo caso, quando si raggiunge un elemento denominato '**relazione**', nel documento XML di origine, viene estratto il valore dell'attributo '**lang**' di questo e inserito in una frase, ignorando qualunque altra cosa che possa riguardare l'elemento in questione.

Quando il documento di destinazione è di tipo XML o HTML, è probabile che si vadano a descrivere dei marcatori che contengono la dichiarazione di attributi. Quando negli attributi si vuole inserire un valore estratto da un nodo del documento XML di origine, non si può usare l'elemento '**xsl:value-of**'; al suo posto si usano delle parentesi graffe, come nell'esempio seguente:

```
<xsl:template match="relazione">
    <HTML LANG="{@lang}">
    <BODY>
    <xsl:apply-templates />
    </BODY>
    </HTML>
</xsl:template>
```

Questa volta, come si può vedere, si va a costruire un documento HTML, dove serve il valore dell'attributo '**lang**' dell'elemento '**relazione**' del file di partenza.

52.2.5 Creazione di componenti XML e HTML

Quando il formato di destinazione della trasformazione è un file XML o HTML, è possibile creare alcune componenti tipiche di questi file con l'ausilio di elementi appositi nel foglio di stile XSLT.

52.2.5.1 Creazione di elementi e di attributi

Quando per qualche ragione è difficile inserire letteralmente il testo che rappresenta i marcatori di un elemento nel documento finale, si può usare nel foglio di stile XSLT l'elemento '**xsl:element**':

```
<xsl:element
    name="nome_elemento"
    [altri_attributi]>
    contenuto
</xsl:element>
```

Per dichiarare un attributo si può usare l'elemento '**xsl:attribute**':

```
<xsl:attribute
    name="nome_attributo"
    [altri_attributi]>
    contenuto
</xsl:element>
```

L'esempio seguente mostra l'uso di questi due elementi, per la costruzione della prima parte di un documento HTML, dove in particolare si vede anche l'uso di '**xsl:value-of**':

```
<xsl:template match="/">
    <xsl:element name="HTML">
        <xsl:attribute name="LANG">
            <xsl:value-of select="relazione/@lang" />
        </xsl:attribute>
        <HEAD>
        <TITLE><xsl:value-of select="relazione/titolo" /></TITLE>
        </HEAD>
        <BODY>
        <xsl:apply-templates />
        </BODY>
    </xsl:element>
</xsl:template>
```

Osservando l'esempio si intende che il documento XML di origine contiene l'elemento '**relazione**' all'inizio della gerarchia; inoltre al suo interno si trova l'elemento '**titolo**' che viene usato per completare l'intestazione del file HTML.

52.2.5.2 Creazione di commenti e di istruzioni di elaborazione

Quando il formato di destinazione è di tipo XML o HTML è possibile inserire dei commenti attraverso l'elemento '**xsl:comment**' nel foglio di stile XSLT:

```
<xsl:comment>
    contenuto
</xsl:comment>
```

In modo analogo, è possibile inserire istruzioni di elaborazione quando il formato di destinazione è di tipo XML, con l'elemento '**xsl:processing-instruction**':

```
<xsl:processing-instruction
    name="nome_istruzione">
    contenuto
</xsl:processing-instruction>
```

Si osservi l'esempio seguente:

```
<xsl:template match="/">
    <xsl:processing-instruction name="xml-stylesheet">href="esempio.css"
    type="text/css"</xsl:processing-instruction>
    <xsl:comment>Ecco un commento</xsl:comment>
    <xsl:apply-templates />
</xsl:template>
```

In questo modo, all'inizio del documento di destinazione si ottiene il testo seguente:

```
<?xml-stylesheet href="esempio.css"
    type="text/css"?>
<!--Ecco un commento-->
```

52.2.6 Xalan

Xalan[1] è un elaboratore XSLT disponibile sia in Java, sia in C++. Qui si fa riferimento all'uso di Xalan-C++, ovvero a un programma compilato in modo da avere un eseguibile che non richiede altre forme di interpretazione:

```
xalan -IN documento_xml_originale -XSL foglio_di_stile ↩
↪     -OUT file_da_generare [altre_opzioni]
```

Il modello sintattico mostrato è più che sufficiente per usare bene Xalan; per le altre opzioni disponibili si può consultare la pagina di manuale *xalan(1)*.

Durante l'elaborazione, Xalan emette alcune informazioni ed eventualmente delle segnalazioni di errore, che dovrebbero tornare utili per correggere il foglio di stile XSLT. Si osservi che quando si procede a una trasformazione che deve generare un documento XML o HTML, Xalan si sofferma anche su errori relativi al formato finale. Per esempio, l'estratto seguente, riferito proprio alla generazione di codice HTML genera un errore a causa della mancata chiusura dell'elemento '**HEAD**' nel file che si ottiene:

```
5   <xsl:template match="/">
6       <HTML>
7       <HEAD>
8       <TITLE><xsl:value-of select="relazione/titolo" /></TITLE>
9       <BODY>
10      <xsl:apply-templates />
11      </BODY>
12  </xsl:element>
```

Supponendo che il file XML originale sia ‘`prova.xml`’, che il foglio di stile XSLT sia contenuto nel file ‘`prova.xsl`’ e che si voglia generare il file ‘`prova.html`’, si dovrebbe procedere con il comando seguente:

```
$ xalan -IN prova.xml -XSL prova.xsl -OUT prova.html [Invio]
```

In questo caso, si ottiene l’errore già descritto:

```
========= Parsing prova.xsl ==========

Fatal Error at (file prova.xsl, line 12, column 18): ↩
↪Expected end of tag 'HEAD'

SAXParseException Message is: Expected end of tag 'HEAD' ↩
↪(prova.xsl, line 12, column 18)
```

I riferimenti ai numeri di riga dell’esempio sono corretti, pertanto si può osservare che gli errori vengono segnalati in posizioni abbastanza lontane rispetto alla loro collocazione effettiva.

52.2.7 Esempio completo

Nelle sezioni seguenti vengono mostrati diversi fogli di stile XSLT per ottenere altrettante trasformazioni a partire da un file XML già mostrato in precedenza. Qui viene riportato nuovamente, con qualche piccola modifica e con l’aggiunta della dichiarazione del DTD incorporata:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE relazione [
    <!ELEMENT relazione     (titolo?, data, contenuto)>
    <!ATTLIST relazione
        xml:lang   CDATA    "">
    <!ELEMENT titolo        (#PCDATA)>
    <!ELEMENT data          (#PCDATA)>
    <!ELEMENT contenuto     (paragrafo+, firma+)>
    <!ELEMENT paragrafo     (#PCDATA)>
    <!ELEMENT firma         (#PCDATA)>
]>

<relazione lang="it">
<titolo>Relazione introduttiva su XML</titolo>

<data>11/11/2011</data>

<contenuto>

<paragrafo>XML sta per Extensible Markup Language.    bla
bla bla... Perché,... così,... perciò,...
sarà...</paragrafo>

<paragrafo>Bla, bla, bla....</paragrafo>

<firma>Pinco Pallino</firma>

</contenuto>
</relazione>
```

Si osservi che questo file, come si vede dalla dichiarazione iniziale, deve usare la codifica UTF-8; di conseguenza, le lettere accentate utilizzano più di un byte per essere rappresentate.

52.2.7.1 Trasformazione in un sorgente LaTeX

Per la trasformazione in un sorgente LaTeX si deve utilizzare un foglio di stile XSLT che elabora il risultato in modalità testo:


```
<xsl:stylesheet
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    version="1.0">
    <xsl:output method="text" encoding="UTF-8" />
    <xsl:template match="/">
\documentclass{article}
\usepackage[utf8]{inputenc}
\usepackage[italian,english]{babel}
\begin{document}
            <xsl:apply-templates />
\end{document}
        </xsl:template>
        <xsl:template match="titolo">
\section{<xsl:apply-templates />}
    </xsl:template>
</xsl:stylesheet>
```

Si può osservare che non sono state stabilite delle regole di trasformazione per gli elementi ‘**`data`**’, ‘**`paragrafo`**’ e ‘**`firma`**’, perché allo scopo risultano sufficienti le regole predefinite. Inoltre, le parentesi graffe sono usate fuori dal contesto in cui servono per ottenere il valore di qualcosa, pertanto hanno soltanto un significato letterale nell’ambito della trasformazione. Ecco cosa si ottiene:

```
\documentclass{article}
\usepackage[utf8]{inputenc}
\usepackage[italian,english]{babel}
\begin{document}


\section{Relazione introduttiva su XML}


11/11/2011



XML sta per Extensible Markup Language.    bla
bla bla... Perché,... così,... perciò,...
sarà...

Bla, bla, bla....

Pinco Pallino



\end{document}
```

Il file che si ottiene utilizza la codifica UTF-8, cosa che consente di evitare l’uso di comandi particolari per rappresentare le lettere accentate, pertanto si è reso necessario l’utilizzo di un «pacchetto» adatto allo scopo (‘**`\usepackage[utf8]{inputenc}`**’). Tuttavia rimangono da risolvere altri problemi legati a caratteri che non possono essere inseriti letteralmente, come per esempio nel caso di ‘**`%`**’, che per LaTeX costituisce l’inizio di un commento. Per questo occorrerebbe inserire nel DTD la dichiarazione di una serie di entità generali, che poi devono essere usate nel sorgente XML.

Infine, si noti che, per semplicità, nella trasformazione viene ignorato completamente il linguaggio, ovvero l’attributo ‘**`lang`**’ dell’elemento ‘**`relazione`**’.

52.2.7.2 Trasformazione in un sorgente HTML

Per la trasformazione in un sorgente HTML si deve utilizzare un foglio di stile XSLT che elabora il risultato in modalità HTML, avendo cura, possibilmente, di predisporre anche la dichiarazione del DTD:

```
<xsl:stylesheet
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    version="1.0">
    <xsl:output
        method="html"
        doctype-public="ISO/IEC 15445:2000//DTD HTML//EN"
        encoding="ISO-8859-1" />
    <xsl:template match="/">
        <xsl:element name="HTML">
            <xsl:attribute name="LANG">
                <xsl:value-of select="relazione/@lang" />
            </xsl:attribute>
```

```
            <HEAD>
            <TITLE><xsl:value-of select="relazione/titolo" /></TITLE>
            <xsl:text disable-output-escaping="yes">
<![CDATA[<STYLE TYPE="text/css">
<!--
BODY {
    background-color: rgb(255, 255, 255)
}
H1 {
    text-align:       right;
    color:            rgb(230, 100, 180)
}

P.paragrafo {
    width:            auto;
    text-align:       left
}
P.data {
    text-align:       left;
    font-weight:      bold
}
P.firma {
    width:            auto;
    text-align:       left
-->
</STYLE>]]>
            </xsl:text>
            </HEAD>
            <BODY>
            <xsl:apply-templates />
            </BODY>
        </xsl:element>
    </xsl:template>
    <xsl:template match="titolo">
        <H1><xsl:apply-templates /></H1>
    </xsl:template>
    <xsl:template match="data">
        <P CLASS="data"><xsl:apply-templates /></P>
    </xsl:template>
    <xsl:template match="paragrafo">
        <P CLASS="paragrafo"><xsl:apply-templates /></P>
    </xsl:template>
    <xsl:template match="firma">
        <P CLASS="firma"><xsl:apply-templates /></P>
    </xsl:template>
</xsl:stylesheet>
```

Questa volta il foglio di stile XSLT è molto più articolato, anche perché incorpora la dichiarazione di alcuni stili CSS. Ecco il risultato che si ottiene; si osservi che il file viene generato usando la codifica ISO-8859-1:

```
<!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
<HTML LANG="it">
<HEAD>
<META http-equiv="Content-Type" content="text/html;
      charset=ISO-8859-1">
<TITLE>Relazione introduttiva su XML</TITLE>
<STYLE TYPE="text/css">
<!--
BODY {
    background-color: rgb(255, 255, 255)
}
H1 {
    text-align:       right;
    color:            rgb(230, 100, 180)
}

P.paragrafo {
    width:            auto;
    text-align:       left
}
P.data {
    text-align:       left;
    font-weight:      bold
}
P.firma {
    width:            auto;
    text-align:       left
-->
</STYLE>
            </HEAD>
<BODY>
<H1>Relazione introduttiva su XML</H1>

<P CLASS="data">11/11/2011</P>



<P CLASS="paragrafo">XML sta per Extensible Markup Language.    bla
bla bla... Perch&eacute;,... cos&igrave;,... perci&ograve;,...
```



```
sar&agrave;...</P>

<P CLASS="paragrafo">Bla, bla, bla....</P>

<P CLASS="firma">Pinco Pallino</P>


</BODY>
</HTML>
```

52.2.7.3 Trasformazione in un sorgente XHTML

Per la trasformazione in un sorgente XHTML si deve utilizzare un foglio di stile XSLT che elabora il risultato in modalità XML:

```
<xsl:stylesheet
    version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns="http://www.w3.org/1999/xhtml">
    <xsl:output
        method="xml"
        encoding="UTF-8" />
    <xsl:template match="/">
        <xsl:element name="html">
            <xsl:attribute name="xml:lang">
                <xsl:value-of select="relazione/@lang" />
            </xsl:attribute>
            <head>
            <title><xsl:value-of select="relazione/titolo" /></title>
            <xsl:text disable-output-escaping="yes">
<![CDATA[<style type="text/css">
<!--
body {
    background-color: rgb(255, 255, 255)
}
h1 {
    text-align:       right;
    color:            rgb(230, 100, 180)
}

p.paragrafo {
    width:            auto;
    text-align:       left
}
p.data {
    text-align:       left;
    font-weight:      bold
}
p.firma {
    width:            auto;
    text-align:       left
-->
</style>]]>
            </xsl:text>
            </head>
            <body>
            <xsl:apply-templates />
            </body>
        </xsl:element>
    </xsl:template>
    <xsl:template match="titolo">
        <h1><xsl:apply-templates /></h1>
    </xsl:template>
    <xsl:template match="data">
        <p class="data"><xsl:apply-templates /></p>
    </xsl:template>
    <xsl:template match="paragrafo">
        <p class="paragrafo"><xsl:apply-templates /></p>
    </xsl:template>
    <xsl:template match="firma">
        <p class="firma"><xsl:apply-templates /></p>
    </xsl:template>
</xsl:stylesheet>
```

Il risultato che si ottiene manca di alcuni incolonnamenti che nelle altre situazioni venivano mantenuti. Per ovviare all'inconveniente, si può provare ad aggiungere l'attributo '**`indent="yes"`**' nell'elemento '**`xsl:output`**':

```
<xsl:stylesheet
    version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns="http://www.w3.org/1999/xhtml">
    <xsl:output
        method="xml"
        indent="yes"
        encoding="UTF-8" />
    ...
```

Ecco il risultato, incolonnato, che si può ottenere:

```
<?xml version="1.0" encoding="UTF-8"?>
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="it">
<head>
<title>Relazione introduttiva su XML</title>
<style type="text/css">
<!--
body {
    background-color: rgb(255, 255, 255)
}
h1 {
    text-align:       right;
    color:            rgb(230, 100, 180)
}

p.paragrafo {
    width:            auto;
    text-align:       left
}
p.data {
    text-align:       left;
    font-weight:      bold
}
p.firma {
    width:            auto;
    text-align:       left
-->
</style>
            </head>
<body>
<h1>Relazione introduttiva su XML</h1>

<p class="data">11/11/2011</p>



<p class="paragrafo">XML sta per Extensible Markup Language.    bla
bla bla... Perché,... così,... perciò,...
sarà...</p>

<p class="paragrafo">Bla, bla, bla....</p>

<p class="firma">Pinco Pallino</p>


</body>
</html>
```

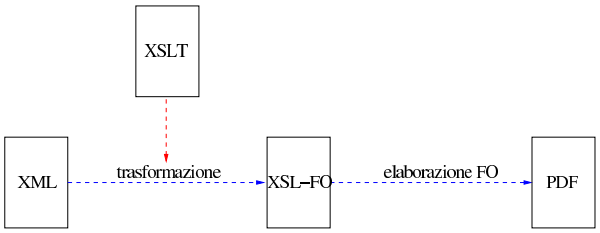
Si ricorda che in questo caso la codifica del file generato è UTF-8, inoltre, si fa notare che nel foglio di stile XSLT appare un dominio applicativo predefinito, allo scopo di riportarlo nel file XHTML di destinazione.

52.3 Introduzione a XSL-FO

XSL-FO (*XSL formatting object*) è un linguaggio realizzato in forma di file XML, con il quale si definisce l'aspetto finale di un documento. In pratica assomiglia concettualmente a HTML o a XHTML, con la differenza che vengono specificati in modo abbastanza preciso i dettagli sulla sua impaginazione per la stampa e generalmente sul suo aspetto finale.

XSL-FO nasce per essere utilizzato prevalentemente nell'ambito di un foglio di stile XSLT, con il quale si definisce la trasformazione di un documento XML in un file XSL-FO. Teoricamente, un navigatore dovrebbe essere in grado di utilizzare direttamente un foglio di stile XSLT che definisce questo tipo di trasformazione; in pratica, inizialmente si esegue la trasformazione per ottenere un file XSL-FO, il quale poi viene convertito in un formato finale comune (di solito è il formato PDF).

Figura 52.55. Passaggi normali per l'utilizzo di XSL-FO.



Nelle sezioni successive vengono mostrati prevalentemente esempi di file scritti secondo il formato XSL-FO, senza usare un foglio di stile XSLT, utilizzando FOP come programma per l'elaborazione, allo scopo di produrre il formato finale per la stampa.

52.3.1 Preparazione degli strumenti

Per poter ottenere la composizione finale di un file in formato XSL-FO occorre un elaboratore FO che generi un altro file più adatto alla consultazione e alla stampa. Qui viene mostrato l'uso di FOP,[2] che assieme a Xalan fa parte del progetto XML di Apache.

A differenza di Xalan, FOP è disponibile solo come programma Java, da interpretare anche una volta compilato. Gli esempi mostrati in questo capitolo sono stati verificati con FOP, eseguito attraverso Kaffe[3].

Una volta installato FOP in un sistema GNU, dovrebbe essere disponibile uno script che consente di avviare il programma senza difficoltà:

```
fop [opzioni] [-fo|-xml] file_ingresso [-xsl file_xsl] ↩
↪ [-pdf|-txt|-altro_formato] file_uscita
```

Se non fosse disponibile questo script, lo si potrebbe riprodurre facilmente utilizzando un contenuto simile a quello seguente:

```
#!/bin/sh
FOPPATH="/usr/share/java/fop.jar"
FOPPATH="$FOPPATH:/usr/share/java/xalan2.jar"
FOPPATH="$FOPPATH:/usr/share/java/xerces.jar"
FOPPATH="$FOPPATH:/usr/share/java/logkit.jar"
FOPPATH="$FOPPATH:/usr/share/java/avalon-framework.jar"
FOPPATH="$FOPPATH:/usr/share/java/batik.jar"
FOPPATH="$FOPPATH:/usr/share/java/jimi-1.0.jar"
CLASSPATH="$FOPPATH:$CLASSPATH"
export CLASSPATH

java $JAVA_OPTS org.apache.fop.apps.Fop "$@"
```

Come si può intendere, in questo caso si considera che i file Java di FOP e di Xalan siano contenuti nella directory '`/usr/share/java/`'; inoltre, viene considerato il contenuto della variabile di ambiente ***JAVA_OPTS*** per le opzioni da passare all'interprete Java. Infine, per quanto riguarda le prove effettuate, '**`java`**' è in realtà un collegamento all'eseguibile che svolge il compito di interpretazione dei programmi Java (per esempio '`/usr/lib/kaffe/bin/java`', oppure '`/usr/bin/kaffe`').

Per verificare subito il funzionamento di FOP si può provare con il sorgente XSL-FO seguente:

```
<?xml version="1.0" encoding="UTF-8"?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
    <fo:layout-master-set>
        <fo:simple-page-master master-name="prova">
            <fo:region-body />
        </fo:simple-page-master>
    </fo:layout-master-set>
    <fo:page-sequence master-reference="prova">
        <fo:flow flow-name="xsl-region-body">
            <fo:block>
                Ciao a tutti.
            </fo:block>
        </fo:flow>
    </fo:page-sequence>
</fo:root>
```

Supponendo che si tratti del file '`prova.fo`', si può generare la sua composizione con il comando seguente:

```
$ fop -fo prova.fo -pdf prova.pdf [ invio ]
```

Utilizzando Kaffe come interprete Java è probabile che appaiano molti errori non gravi; a parte questi, ciò che si dovrebbe vedere durante la composizione sono i messaggi seguenti:

```
[INFO] FOP 0.20.4
[INFO] building formatting object tree
[INFO] [1]
[INFO] Parsing of document complete, stopping renderer
```

Se non si arriva a vedere la frase finale in cui si conferma la conclusione dell'analisi e della composizione, qualcosa è andato storto. Quello che si deve ottenere è quindi il file '`prova.pdf`', contenente la scritta «Ciao a tutti.», collocata a partire dall'angolo superiore sinistro del foglio, senza alcun margine (il foglio dovrebbe avere il formato Lettera, ovvero 8,5 in × 11 in).

A parte la facilità con cui è stato utilizzato FOP per questo esempio, si possono presentare problemi difficili da comprendere, perché FOP è un progetto ancora all'inizio del suo sviluppo e non genera informazioni sufficienti a trovare gli errori sintattici nel sorgente XSL-FO. Per esempio, nel momento in cui si scrive questo capitolo, se si commette un piccolo errore come quello seguente (alla quarta riga, il marcatore iniziale dell'elemento '**`fo:simple-page-master`**' non è concluso dal simbolo '**`>`**'), i messaggi di XSL-FO non aiutano a capirlo:

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
3      <fo:layout-master-set>
4          <fo:simple-page-master master-name="prova"
5              <fo:region-body />
6          </fo:simple-page-master>
7      </fo:layout-master-set>
8      <fo:page-sequence master-reference="prova">
9          <fo:flow flow-name="xsl-region-body">
10             <fo:block>
11                 Ciao a tutti.
12             </fo:block>
13         </fo:flow>
14     </fo:page-sequence>
15 </fo:root>
```

```
$ fop -fo prova.fo -pdf prova.pdf [invio]

[INFO] FOP 0.20.4
[INFO] building formatting object tree
[ERROR] Can't find bundle for base name ↩
↪org.apache.xerces.impl.msg.XMLMessages, locale en_US
```

Date le difficoltà, può essere conveniente l'abbinamento con un DTD, da verificare prima della composizione; eventualmente si può usare quello presentato nella sezione 52.3.4. Purtroppo il DTD non può verificare tutti i vincoli reali del formato XSL-FO, ma almeno consente di evitare errori grossolani.

Se si salva il file del DTD con il nome '`fo.dtd`', nella directory corrente, si può modificare l'intestazione dei file XSL-FO nel modo seguente:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE fo:root SYSTEM "fo.dtd">
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
...
</fo:root>
```

Per l'analisi sintattica si può usare il pacchetto SP di James Clark (come descritto nella sezione 51.2), ma prima occorre procurarsi anche una dichiarazione XML standard assieme al suo catalogo. Si suppone di avere così anche i file '`xml.dcl`' e '`xml.cat`' (si riveda eventualmente quanto spiegato all'inizio del capitolo), entrambi collocati nella directory corrente, assieme al DTD. Naturalmente il file '`xml.cat`' deve contenere un riferimento al file '`xml.dcl`' nella directory corrente:

```
SGMLDECL "xml.dcl"
```

Per avviare la verifica del file '`prova.fo`' che contiene la dichiarazione corretta del DTD, si può procedere con il comando seguente:

```
$ cat prova.fo | nsgmls -s -c ./xml.cat [invio]
```

Data questa premessa, negli esempi proposti di file XSL-FO viene inserita sempre l'intestazione che dichiara il DTD (salvo eccezioni), anche se si tratta di un'informazione superflua per la composizione.

52.3.2 Struttura generale del documento secondo XSL-FO

Un foglio di stile XSL-FO è un documento XML abbinato al dominio applicativo (*namespace*) *http://www.w3.org/1999/XSL/Format* , a cui si associa normalmente il prefisso '**`fo:`**'. L'elemento principale di questo documento, ovvero quello che contiene tutti gli altri, è '**`fo:root`**':

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE fo:root SYSTEM "fo.dtd">
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
    ...
</fo:root>
```

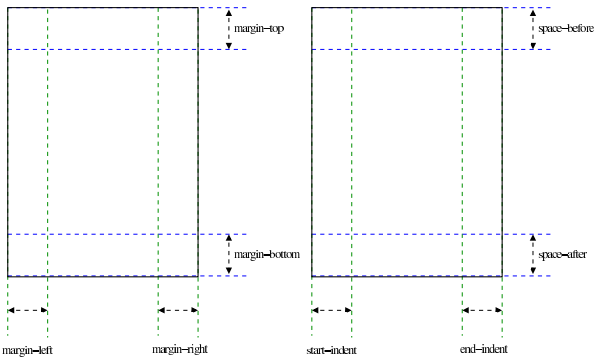
Questo elemento contiene obbligatoriamente '**`fo:layout-master-set`**' e uno o più elementi '**`fo:page-sequence`**'. Il primo serve a descrivere la pagina e l'impaginazione generale, mentre il secondo serve a incorporare il contenuto di ciò che si vuole stampare effettivamente (il testo, le immagini, ecc.):

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE fo:root SYSTEM "fo.dtd">
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
    <fo:layout-master-set>
        ...
    </fo:layout-master-set>
    <fo:page-sequence>
        ...
    </fo:page-sequence>
    ...
</fo:root>
```

52.3.3 Margini e direzione

Prima di poter descrivere la struttura della pagina secondo XSL-FO, è necessario comprendere alcuni concetti generali legati alla gestione dei margini: i «blocchi», ovvero gli oggetti che per loro natura possono essere considerati come dei rettangoli, ovvero delle scatole bidimensionali, hanno una gestione comune dei margini. Si osservi la figura 52.65.

Figura 52.65. Margini nelle scatole XSL-FO.



La figura mostra due scatole con margini definiti attraverso gruppi differenti di attributi: '**`margin-top`**', '**`margin-bottom`**', '**`margin-left`**' e '**`margin-right`**'; '**`space-before`**', '**`space-after`**', '**`start-indent`**' e '**`end-indent`**'. Il primo gruppo di attributi riguarda letteralmente il margine superiore, il margine inferiore, il margine sinistro e il margine destro, mentre il secondo gruppo tiene conto della direzione della scrittura. In pratica, la figura mostra l'uso del secondo gruppo di attributi quando la scrittura avviene da sinistra a destra e dall'alto in basso; si può comprendere intuitivamente come cambino significato questi margini se l'andamento della scrittura cambia.

Questi due gruppi di definizione dei margini sono alternativi e se si usano assieme, viene scelto un tipo di margine in base a delle precedenze. Tuttavia, in generale è meglio usare il secondo gruppo di margini che è più generale.

Per quanto riguarda la direzione, questa può essere controllata con l'attributo '**`writing-mode`**', a cui si possono attribuire stringhe come '**`lr-tb`**' e '**`rl-tb`**' (in questo caso riferite rispettivamente alla

scrittura da sinistra a destra e dall'alto in basso, oppure da destra a sinistra e dall'alto in basso). Tuttavia, è probabile che gli strumenti di elaborazione di file XSL-FO, nel loro sviluppo iniziale, non siano in grado di adeguarsi al cambiamento di direzione.

52.3.4 Un DTD per XSL-FO

All'indirizzo *allegati/sgml/fo.dtd* è disponibile un DTD realizzato appositamente durante lo studio che ha preceduto la realizzazione degli esempi che sono proposti in questi capitoli, attingendo dalle informazioni contenute nel documento *xsl:fo short reference*, il quale fa parte della documentazione di FOP, oltre che dal documento *Extensible stylesheet language (XSL)*, presso *http://www.w3.org/TR/xsl/* .

Si osservi che nel DTD manca la possibilità di utilizzare attributi generici riferiti ai bordi (per esempio '**`border`**', '**`border-color`**', ecc.), la presenza dei quali risulterebbe in un errore. Si ritiene che non sia il caso di utilizzare queste proprietà complessive come si può fare con i fogli di stile CSS; tuttavia, non sarebbe difficile aggiungere tali proprietà al DTD se lo si ritiene necessario.

52.4 XSL-FO: impaginazione

L'impaginazione secondo XSL-FO viene definita attraverso dei modelli a cui si fa riferimento per mezzo di un nome. Quando si dichiara il modello, l'elemento relativo utilizza l'attributo '**`master-name`**'; quando vi si fa riferimento, l'elemento utilizza l'attributo '**`master-reference`**'.

52.4.1 Impaginazione semplificata

La definizione minima della pagina e del contenuto si ottiene con una struttura XSL-FO simile a quella seguente, dove però non sono ancora stati annotati gli attributi obbligatori:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE fo:root SYSTEM "fo.dtd">
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
    <fo:layout-master-set>
        <fo:simple-page-master>
        ...
        </fo:simple-page-master>
        ...
        <fo:region-body />
    </fo:layout-master-set>
    <fo:page-sequence>
        <fo:flow>
        ...
        </fo:flow>
        ...
    </fo:page-sequence>
    ...
</fo:root>
```

All'interno di '**`fo:layout-master-set`**' si collocano gli elementi che descrivono le caratteristiche delle pagine. Nella situazione più semplice si usano elementi '**`fo:simple-page-master`**', all'interno dei quali si definiscono delle «regioni»:

```
<fo:simple-page-master
    master-name="nome_tipo_impaginazione"
    [page-height="altezza_pagina"]
    [page-width="ampiezza_pagina"]
    [writing-mode="direzione"]
    [space-before="spazio_prima"]
    [space-after="spazio_dopo"]
    [start-indent="rientro_iniziale"]
    [end-indent="rientro_finale"]
    [altri_attributi]>

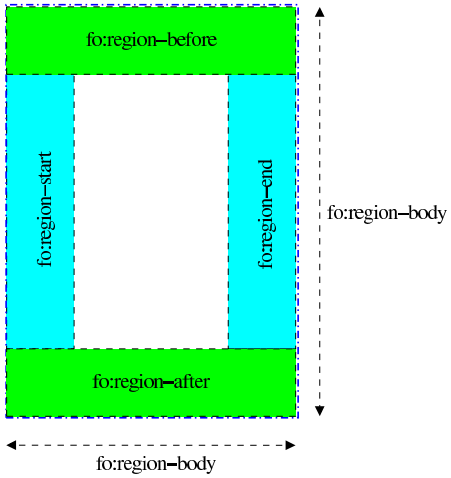
    regioni

</fo:simple-page-master>
```


Il significato degli attributi mostrati nello schema è abbastanza intuitivo (in modo particolare il problema dei margini delle scatole è già descritto nella sezione 52.3), a parte l'attributo obbligatorio '**`master-name`**', che ha lo scopo di dare un nome al tipo di impaginazione che viene dichiarato in quel contesto. Successivamente, quando si descrivono le componenti del corpo della pagina, viene fatto riferimento a questo nome per richiamare il tipo di impaginazione desiderato.

All'interno dei margini fissati con gli attributi appropriati dell'elemento '**`fo:simple-page-master`**' si collocano delle regioni, dichiarate attraverso gli elementi '**`fo:region-before`**', '**`fo:region-after`**', '**`fo:region-start`**', '**`fo:region-end`**' e '**`fo:region-body`**'.

Figura 52.67. Regioni all'interno dei margini di una pagina.



Guardando la figura 52.67, che rappresenta lo spazio interno ai margini di una pagina, si deve osservare che la scatola definita dall'elemento '**`fo:region-body`**' contiene, sovrapposte, le altre quattro regioni. Inoltre, è importante osservare i nomi delle regioni, che rappresentano una collocazione relativa alla direzione della scrittura; per esempio, con una scrittura che procede dal basso verso l'altro, si scambiano di posto le regioni '**`fo:region-before`**' e '**`fo:region-after`**' rispetto alla figura; inoltre, una scrittura che procede da destra verso sinistra fa sì che si scambi la collocazione delle regioni dichiarate dagli elementi '**`fo:region-start`**' e '**`fo:region-end`**'.

```
<fo:region-body
    [space-before="spazio_prima"]
    [space-after="spazio_dopo"]
    [start-indent="rientro_iniziale"]
    [end-indent="rientro_finale"]
    [writing-mode="direzione"]
    [reference-orientation="rotazione"]
    [altri_attributi] />
```

```
<fo:region-{before|after|start|end}
    extent="dimensione"
    [writing-mode="direzione"]
    [reference-orientation="rotazione"]
    [altri_attributi] />
```

Gli elementi che dichiarano le regioni sono vuoti. A esclusione dell'elemento '**`fo:region-body`**', gli altri hanno tutti un attributo obbligatorio che ne definisce l'estensione (l'ampiezza per '**`fo:region-start`**' e '**`fo:region-end`**'; l'altezza per '**`fo:region-before`**' e '**`fo:region-after`**').

Nei modelli sintattici mostrati appare un attributo che non è anco-

ra stato preso in considerazione: '**reference-orientation**'. Con questo è possibile fa ruotare il contenuto, in multipli di 90 gradi, cosa che può risultare particolarmente utile per il contenuto delle regioni laterali.

Si comprende intuitivamente il significato di queste regioni che si trovano ai bordi del corpo della pagina: consentono di inserire delle intestazioni, dei piè di pagina, delle note a margine e simili. Ciò che è importante è rendersi conto che l'area che descrivono si sovrappone al corpo della pagina; pertanto, solitamente il contenuto che si va a inserire nel corpo viene controllato da margini ulteriori.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE fo:root SYSTEM "fo.dtd">
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
    <fo:layout-master-set>
        <fo:simple-page-master
                master-name="normale"
                page-width="210mm"
                page-height="297mm"
                margin-top="2cm"
                margin-bottom="2cm"
                margin-left="2cm"
                margin-right="2cm">
            <fo:region-body   margin-top="1cm"
                              margin-bottom="1cm"/>
            <fo:region-before extent="1cm" />
            <fo:region-after  extent="1cm" />
        </fo:simple-page-master>
        ...
    </fo:layout-master-set>
    <fo:page-sequence>
        ...
    </fo:page-sequence>
    ...
</fo:root>
```

L'esempio che appare sopra mostra l'uso di un elemento '**fo:simple-page-master**', con il quale si dichiara un tipo di impaginazione denominato '**normale**', con le dimensioni di un foglio A4 (21 cm × 29,7 cm), con margini uguali di 2 cm, all'interno dei quali si trova un corpo, che a sua volta ha margini superiori e inferiori, pari all'estensione delle regioni rispettive.

Per riempire effettivamente le pagine di contenuti, occorre intervenire all'interno di elementi '**fo:page-sequence**', con cui si fa riferimento al tipo di impaginazione che si vuole usare:

```
<fo:page-sequence
    master-reference="nome_tipo_impaginazione"
    [initial-page-number="numero_iniziale"]
    [force-page-count="auto"|"odd"|"even"|altro]
    [country="nazionalità_iso_3166"]
    [language="lingua_iso_639"]
    [altri_attributi]>

    contenuti

</fo:page-sequence>
```

Dal modello sintattico si vede che il riferimento al tipo di impaginazione si ottiene con l'attributo '**master-reference**', che pertanto è obbligatorio. Sono inoltre da considerare gli attributi '**initial-number**' e '**force-page-count**': il primo consente di stabilire il numero di pagina iniziale e il secondo permette di stabilire se il gruppo di pagine in questione deve essere complessivamente dispari, pari o se debbano essere rispettate altre regole.

Gli attributi '**country**' e '**language**', nell'insieme servono a definire le caratteristiche locali del testo, stabilendo la nazionalità e il linguaggio. Formalmente, le sigle che si utilizzano per questo sono quelle definite nel documento RFC 1766, che in pratica corrisponde agli standard ISO 3166 (tabella 13.5) e ISO 639 (tabella 13.4).

Un elemento '**fo:page-sequence**' contiene necessariamente l'elemento '**fo:flow**' (uno solo), che a sua volta contiene ciò che viene

distribuito nelle pagine (di solito il testo), come se fosse un flusso di informazioni. L'elemento '**fo:flow**' deve dichiarare, attraverso l'attributo '**flow-name**', in quale regione della pagina si inserisce il flusso in questione:

```
<fo:flow
    flow-name="xsl-region-{body|before|after|start|end}"

    flusso

</fo:flow>
```

L'esempio seguente riprende quanto già presentato a proposito della dichiarazione del tipo di impaginazione, associato a un flusso che riguarda il corpo della pagina stessa:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE fo:root SYSTEM "fo.dtd">
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
    <fo:layout-master-set>
        <fo:simple-page-master
                master-name="normale"
                page-width="210mm"
                page-height="297mm"
                margin-top="2cm"
                margin-bottom="2cm"
                margin-left="2cm"
                margin-right="2cm">
            <fo:region-body   margin-top="1cm"
                              margin-bottom="1cm"/>
            <fo:region-before extent="1cm" />
            <fo:region-after  extent="1cm" />
        </fo:simple-page-master>
    </fo:layout-master-set>
    <fo:page-sequence
            master-reference="normale"
            initial-page-number="1"
            force-page-count="even"
            language="it"
            country="it">
        <fo:flow flow-name="xsl-region-body">
            <fo:block>
                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
                bla bla bla.
            </fo:block>
            <fo:block>
                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
                bla bla bla.
            </fo:block>
        </fo:flow>
    </fo:page-sequence>
</fo:root>
```

Quello che si ottiene dalla trasformazione di questo file XSL-FO sono due pagine (l'ultima è bianca) che contengono il testo che si può vedere, suddiviso in due capoversi. Si può osservare che il margine superiore è di 3 cm, perché 2 cm definiti dall'impaginazione si sommano al margine del corpo della pagina.

Le pagine ottenute sono due perché è stato richiesto espressamente di impiegare una quantità pari (*even*) di pagine con l'attributo '**force-page-content**' dell'elemento '**fo:page-sequence**'.

Se si tralasciano gli attributi, la sintassi completa di '**fo:page-sequence**' corrisponde al modello seguente:

```
<fo:page-sequence attributi>
    [<fo:title attributi>
        titolo
    </fo:title>]
    [<fo:static-content flow-name="regione">
        contenuto_statico
    </fo:static-content>]
    …
    <fo:flow flow-name="regione">
        flusso
    </fo:flow>
</fo:page-sequence>
```

L'elemento '**`fo:title`**' consente di attribuire un titolo, che non fa parte, necessariamente, del risultato della composizione. Gli elementi '**`fo:static-content`**' si comportano in modo simile a '**`fo:flow`**', per quanto riguarda la selezione della regione di competenza, ma servono per definire del testo che si vuole appaia su tutte le pagine (come può essere per una riga di intestazione o alla base della pagina).

L'esempio seguente espande quanto già visto in quello precedente, aggiungendo un'intestazione e una riga a piè pagina, dove, in particolare, viene collocato anche il numero della pagina. Ovviamente, per poter apprezzare il fatto che l'intestazione e il piè di pagina rimangono costanti, occorre mettere qualcosa di più nel corpo:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE fo:root SYSTEM "fo.dtd">
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
    <fo:layout-master-set>
        <fo:simple-page-master
                master-name="normale"
                page-width="210mm"
                page-height="297mm"
                margin-top="2cm"
                margin-bottom="2cm"
                margin-left="2cm"
                margin-right="2cm">
            <fo:region-body   margin-top="1cm"
                              margin-bottom="1cm"/>
            <fo:region-before extent="1cm" />
            <fo:region-after  extent="1cm" />
        </fo:simple-page-master>
    </fo:layout-master-set>
    <fo:page-sequence
            master-reference="normale"
            initial-page-number="1"
            force-page-count="even"
            language="it"
            country="it">
        <fo:static-content flow-name="xsl-region-before">
            <fo:block>
                Un esempio per cominciare.
            </fo:block>
        </fo:static-content>
        <fo:static-content flow-name="xsl-region-after">
            <fo:block>
                pagina <fo:page-number />
            </fo:block>
        </fo:static-content>
        <fo:flow flow-name="xsl-region-body">
            <fo:block>
                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
                bla bla bla.
            </fo:block>
            <!-- Inserire qui altri fo:block per ottenere
                 più pagine -->
            <fo:block>
                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
                bla bla bla.
            </fo:block>
        </fo:flow>
    </fo:page-sequence>
</fo:root>
```

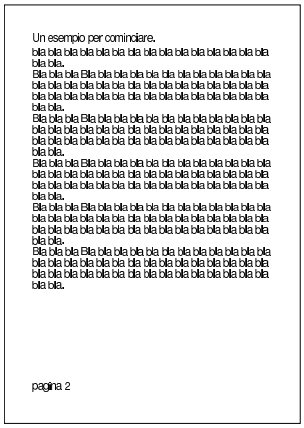

Si osservi che gli esempi devono essere salvati in file con codifica UTF-8, altrimenti diventa indispensabile usare delle entità che facciano riferimento al punto di codifica, quando si usano simboli al di sopra di U+007F. Pertanto, se necessario, anche il commento va scritto nel modo corretto, per non creare problemi al programma che lo elabora:

```
<!-- Inserire qui altri fo:block per ottenere
     pi&#x00F9; pagine -->
```

Oppure:

```
<!-- Inserire qui altri fo:block per ottenere
     piu' pagine -->
```

Figura 52.73. L'aspetto di una pagina ottenuta dalla composizione dell'esempio. Si osservi che i caratteri sono stati ingranditi rispetto a quanto si otterrebbe normalmente, considerando le proporzioni.



52.4.2 Impaginazione più articolata

Nella sezione precedente è stata mostrata la possibilità di richiedere espressamente una quantità pari o dispari di pagine per una certa sequenza di pagine. Tuttavia questo non basta per controllare l'estetica di un bel libro, perché di solito si vuole che le pagine cambino impostazione in modo automatico all'inizio di un capitolo. Per raggiungere risultati come questo, al posto di usare l'elemento '**`fo:simple-page-master`**', si usa piuttosto '**`fo:page-sequence-master`**', che però si avvale in pratica delle dichiarazioni di altri elementi '**`fo:simple-page-master`**':

```
<fo:page-sequence-master
    master-name="nome_tipo_impaginazione">
    [<fo:single-page-master-reference
        master-reference=nome_tipo_impaginazione />]
    [<fo:repeatable-page-master-reference
        master-reference=nome_tipo_impaginazione
        [maximum-repeat=max_ripetizioni] />]
    [<fo:repeatable-page-master-alternatives
        [maximum-repeat=max_ripetizioni]>
        <fo:conditional-page-master-reference
            master-reference="nome_tipo_impaginazione"
            page-position="posizione"
            [altri_attributi] />
        …
    </fo:repeatable-page-master-alternatives>]
    …
</fo:page-sequence-master>
```

Inizialmente, l'insieme è piuttosto complesso. L'elemento più esterno, '**fo:page-sequence-master**' definisce un tipo di impaginazione a cui in seguito è possibile fare riferimento. All'interno di questo elemento possono apparire tre tipi di elementi, anche ripetutamente, con lo scopo di descrivere in sequenza i tipi di pagina da usare.

L'elemento '**fo:single-page-master-reference**', che si riferisce a un certo tipo di pagina in base all'attributo '**master-reference**', stabilisce che deve apparire una sola pagina con quelle caratteristiche. L'elemento '**fo:repeatable-page-master-reference**' stabilisce di utilizzare un certo tipo di pagina, attraverso l'attributo '**master-reference**', per una quantità imprecisata di pagine (salvo l'uso dell'attributo '**maximum-repeat**'. Si osservi l'esempio seguente:

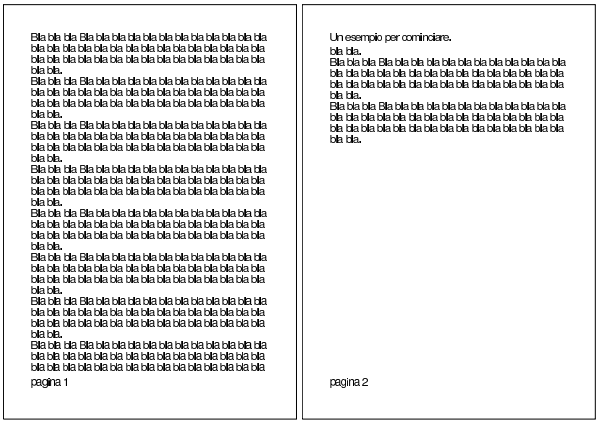
```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <!DOCTYPE fo:root SYSTEM "fo.dtd">
3  <fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
4      <fo:layout-master-set>
5          <fo:simple-page-master
6              master-name="normale"
7              page-width="210mm"
8              page-height="297mm"
9              margin-top="2cm"
10             margin-bottom="2cm"
11             margin-left="2cm"
12             margin-right="2cm">
13             <fo:region-body
14                 margin-top="1cm"
15                 margin-bottom="1cm"/>
16             <fo:region-before extent="1cm" />
17             <fo:region-after  extent="1cm" />
18         </fo:simple-page-master>
19         <fo:simple-page-master
20             master-name="speciale"
21             page-width="210mm"
22             page-height="297mm"
23             margin-top="2cm"
24             margin-bottom="2cm"
25             margin-left="2cm"
26             margin-right="2cm">
27             <fo:region-body
28                 margin-bottom="1cm"/>
29             <fo:region-after
30                 extent="1cm" />
31         </fo:simple-page-master>
32         <fo:page-sequence-master
33             master-name="capitolo">
34             <fo:single-page-master-reference
35                 master-reference="speciale" />
36             <fo:repeatable-page-master-reference
37                 master-reference="normale" />
38         </fo:page-sequence-master>
39     </fo:layout-master-set>
40     <fo:page-sequence
41             master-reference="capitolo"
42             initial-page-number="1"
43             force-page-count="even"
44             language="it"
45             country="it">
46         <fo:static-content
47             flow-name="xsl-region-before">
48             <fo:block>
49                 Un esempio per cominciare.
50             </fo:block>
51         </fo:static-content>
52         <fo:static-content
53             flow-name="xsl-region-after">
54             <fo:block>
55                 pagina <fo:page-number />
56             </fo:block>
57         </fo:static-content>
58         <fo:flow flow-name="xsl-region-body">
59             <fo:block>
60                 Bla bla bla Bla bla bla bla bla bla bla
61                 bla bla bla bla bla bla bla bla bla bla
62                 bla bla bla bla bla bla bla bla bla bla
63                 bla bla bla bla bla bla bla bla bla bla
```



```
64                 bla bla bla bla bla bla bla.
65             </fo:block>
66             <!-- Inserire qui altri fo:block per
67                  ottenere più pagine -->
68             <fo:block>
69                 Bla bla bla Bla bla bla bla bla bla bla
70                 bla bla bla bla bla bla bla bla bla bla
71                 bla bla bla bla bla bla bla bla bla bla
72                 bla bla bla bla bla bla bla bla bla bla
73                 bla bla bla bla bla bla bla.
74             </fo:block>
75         </fo:flow>
76     </fo:page-sequence>
77 </fo:root>
```

Questa volta vengono dichiarati due tipi di impaginazione «semplice»; una denominata '**normale**', l'altra denominata '**speciale**' (righe da 5 a 31). La differenza tra i due tipi di impaginazione sta nell'assenza, nel secondo caso, della regione superiore, con la conseguente assenza del margine superiore nel corpo. In questo modo, si vuole fare sì che un capitolo utilizzi come prima pagina questo secondo tipo di impaginazione, senza usare la riga di intestazione. Pertanto, si definisce un tipo di impaginazione più complesso, denominato proprio '**capitolo**', che prevede come prima pagina l'impaginazione '**speciale**' e come pagine successive l'impaginazione '**normale**'. Per quanto riguarda l'elemento '**fo:page-sequence**' nulla è cambiato rispetto all'esempio precedente.

Figura 52.75. L'aspetto delle prime due pagine ottenute dalla composizione dell'esempio. Si osservi che i caratteri sono stati ingranditi rispetto a quanto si otterrebbe normalmente, considerando le proporzioni.



Per ottenere effetti più complessi si può usare l'elemento '**fo:repeatable-page-master-alternatives**', che contiene necessariamente elementi '**fo:conditional-page-master-referenc**

```
<fo:repeatable-page-master-alternatives
        [maximum-repeat=max_ripetizioni]>
    <fo:conditional-page-master-reference
        master-reference="nome_tipo_impaginazione"
        page-position="first"|"last"|"rest"|"any"
        [altri_attributi] />
    ...
</fo:repeatable-page-master-alternatives>
```

In pratica, attraverso gli elementi '**conditional-page-master-refere** si vanno a individuare dei sottogruppi di pagine in base a una condizione. La sintassi mostra la selezione più semplice, attraverso l'attributo '**page-position**', con il quale si può individuare la prima pagina, l'ultima, le pagine successive alla prima oppure qualunque pagina. Tuttavia sono disponibili altri attributi per condizioni più sofisticate, che qui non vengono descritti. L'esempio seguente svolge lo stesso lavoro di quello precedente, con l'uso di queste selezioni condizionali:

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <!DOCTYPE fo:root SYSTEM "fo.dtd">
3  <fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
4      <fo:layout-master-set>
5          <fo:simple-page-master
6              master-name="normale"
7              page-width="210mm"
8              page-height="297mm"
9              margin-top="2cm"
10             margin-bottom="2cm"
11             margin-left="2cm"
12             margin-right="2cm">
13             <fo:region-body margin-top="1cm"
14                 margin-bottom="1cm"/>
15             <fo:region-before extent="1cm" />
16             <fo:region-after  extent="1cm" />
17         </fo:simple-page-master>
18         <fo:simple-page-master
19             master-name="speciale"
20             page-width="210mm"
21             page-height="297mm"
22             margin-top="2cm"
23             margin-bottom="2cm"
24             margin-left="2cm"
25             margin-right="2cm">
26             <fo:region-body margin-bottom="1cm"/>
27             <fo:region-after extent="1cm" />
28         </fo:simple-page-master>
29         <fo:page-sequence-master master-name="capitolo">
30             <fo:repeatable-page-master-alternatives>
31                 <fo:conditional-page-master-reference
32                     master-reference="speciale"
33                     page-position="first" />
34                 <fo:conditional-page-master-reference
35                     master-reference="normale"
36                     page-position="rest" />
37             </fo:repeatable-page-master-alternatives>
38         </fo:page-sequence-master>
39     </fo:layout-master-set>
40     <fo:page-sequence
41         master-reference="capitolo"
42         initial-page-number="1"
43         force-page-count="even"
44         language="it"
45         country="it">
46         <fo:static-content
47             flow-name="xsl-region-before">
48             <fo:block>
49                 Un esempio per cominciare.
50             </fo:block>
51         </fo:static-content>
52         <fo:static-content
53             flow-name="xsl-region-after">
54             <fo:block>
55                 pagina <fo:page-number />
56             </fo:block>
57         </fo:static-content>
58         <fo:flow flow-name="xsl-region-body">
59             <fo:block>
60                 Bla bla bla Bla bla bla bla bla bla bla
61                 bla bla bla bla bla bla bla bla bla bla
62                 bla bla bla bla bla bla bla bla bla bla
63                 bla bla bla bla bla bla bla bla bla bla
64                 bla bla bla bla bla bla bla.
65             </fo:block>
66             <!-- Inserire qui altri fo:block per
67                  ottenere più pagine -->
68             <fo:block>
69                 Bla bla bla Bla bla bla bla bla bla bla
70                 bla bla bla bla bla bla bla bla bla bla
71                 bla bla bla bla bla bla bla bla bla bla
72                 bla bla bla bla bla bla bla bla bla bla
73                 bla bla bla bla bla bla bla.
74             </fo:block>
75         </fo:flow>
76     </fo:page-sequence>
77 </fo:root>
```

La differenza rispetto all'esempio precedente si trova precisamente nelle righe da 29 a 38.

52.4.3 Contenuto

A proposito del contenuto del documento, è già stato mostrato l'uso e la struttura successiva all'elemento '**`fo:page-sequence`**':

```
<fo:page-sequence
        master-reference="nome_tipo_impaginazione"
        [initial-page-number="numero_iniziale"]
        [force-page-count="auto"|"odd"|"even"|altro]
        [country="nazionalità_iso_3166"]
        [language="lingua_iso_639"]
        [altri_attributi]>
    [<fo:title attributi>
        titolo
    </fo:title>]
    [<fo:static-content flow-name="regione">
        contenuto_statico
    </fo:static-content>]
    …
    <fo:flow flow-name="regione">
        flusso
    </fo:flow>
</fo:page-sequence>
```

L'elemento '**`fo:page-sequence`**' rappresenta letteralmente una sequenza di pagine, come potrebbe essere un capitolo o una parte di un libro, dove, tra le altre cose, può essere utile ripartire con una nuova numerazione delle pagine. Pertanto, un documento può avere più elementi '**`fo:page-sequence`**' distinti.

Gli elementi '**`fo:flow`**', o '**`fo:static-content`**', rappresentano in pratica la destinazione del contenuto, che va inserito nella regione appropriata. Questi elementi possono contenere dei «blocchi», o più precisamente elementi '**`fo:block`**', '**`fo:block-container`**', '**`fo:list-block`**', '**`fo:table`**', '**`fo:table-and-caption`**'.

L'elemento '**`fo:block`**' ha la particolarità di contenere sia testo lineare, sia altri blocchi (quindi anche se stesso). Pertanto viene usato indifferentemente come un equivalente dell'elemento '**`DIV`**' o dell'elemento '**`P`**' di HTML.

52.5 XSL-FO: contenuto

Il contenuto di una regione del corpo definita dall'impaginazione, è fatto sommariamente di blocchi, come l'elemento '**`fo:block`**', e di componenti lineari, in base al contesto. Generalmente, un blocco non può essere contenuto in un contesto lineare, a meno che questo sia incorporato in un elemento apposito, come '**`fo:inline-container`**'.

Un blocco è idealmente un oggetto a due dimensioni, rettangolare, per il quale potrebbero essere previsti dei margini su tutti i lati; un componente lineare è invece inteso come un oggetto monodimensionale che può avere margini solo prima e dopo.

In generale, gli elementi che descrivono un blocco o una componente lineare condividono delle opzioni comuni per definire i margini, il bordo, il colore di fondo e altre caratteristiche. La tabella 52.77 elenca alcuni attributi abbastanza comuni.

Tabella 52.77. Alcuni attributi abbastanza comuni negli elementi che contengono del testo, sia in forma di blocco, sia in forma lineare.

Attributo	Descrizione
`space-start="`*spazio_iniziale*`"` `space-end="`*spazio_finale*`"`	Riguarda i componenti lineari e richiede uno spazio iniziale o finale.

Attributo	Descrizione

`*before*="`***prima***`"`
`*after*="`***dopo***`"`
`*start*="`***inizio***`"`
`*end*="`***fine***`"`
	Un attributo che contiene le parole chiave evidenziate rappresenta una proprietà relativa alla posizione rispettiva, tenendo conto della direzione di scrittura.

`space-before="`***spazio_prima***`"`
`space-after="`***spazio_dopo***`"`
`start-indent="`***rientro_iniziale***`"`
`end-indent="`***rientro_finale***`"`
	Riguarda i componenti a blocchi e definisce i margini.

`border-before-color="`***colore***`"`
`border-after-color="`***colore***`"`
`border-start-color="`***colore***`"`
`border-end-color="`***colore***`"`
	Definisce il colore del bordo.

`border-before-style="`***stile***`"`
`border-after-style="`***stile***`"`
`border-start-style="`***stile***`"`
`border-end-style="`***stile***`"`
	Definisce lo stile del bordo. Possono essere usate parole chiave come: '**`none`**', '**`dotted`**', '**`dashed`**', '**`solid`**', '**`double`**'.

`border-before-width="`***spessore***`"`
`border-after-width="`***spessore***`"`
`border-start-width="`***spessore***`"`
`border-end-width="`***spessore***`"`
	Definisce lo spessore del bordo.

`font-family="`***nome***`"` `font-family="serif"`↩ ↪`|"sans-serif"`↩ ↪`|"cursive"|"fantasy"`↩ ↪`|"monospace"`	Dichiara il nome di un carattere tipografico. Il nome può essere indicato in modo preciso, oppure attraverso parole chiave che si riferiscono alla caratteristica di massima.

`font-size="`***altezza***`"`
`font-size="xx-small"`↩
↪`|"x-small"|"small"`↩
↪`|"medium"|"large"|"x-large"`↩
↪`|"xx-large"`
`font-size="larger"|"smaller"`
`font-size="`***n***`%"`
`font-size="`***n***`em"`
	Dichiara la dimensione del carattere tipografico. La dimensione può essere data con un valore assoluto, espresso attraverso un numero e la sua unità di misura; un valore assoluto specificato attraverso una parola chiave; un valore relativo specificato attraverso una parola chiave; un valore relativo specificato in modo percentuale o in quadratoni («em»).

`font-style="normal"`↩ ↪`|"italic"|"oblique"`↩ ↪`|"backslant"`	Dichiara stile del carattere, inteso come la sua inclinazione o la sua forma corsiva.
`font-stretch="ultra-condensed"`↩ ↪`|"extra-condensed"|"condensed"`↩ ↪`|"semi-condensed"|"normal"`↩ ↪`|"semi-expanded"|"expanded"`↩ ↪`|"extra-condensed"`↩ ↪`|"ultra-expanded"`	Dichiara la larghezza del carattere.
`font-variant="normal"|"small-caps"`	Dichiara l'utilizzo o meno del maiuscoletto.

Attributo	Descrizione
`font-weight="normal"|"bold"`↩ ↪`|"bolder"`↩ ↪`|"lighter"`	Dichiara lo spessore del carattere (normale, nero, nerissimo, chiaro).
`line-height="normal"|"`***altezza***`"`↩ ↪`|"`***n***`"|"`***n***`%"`	Dichiara l'altezza della riga. Se si indica un numero privo di unità di misura, si fa riferimento alla dimensione del carattere (***n*** volte l'altezza del carattere); se si indica una percentuale, questa si riferisce all'altezza del carattere.
`text-align="start"|"center"|"end"`↩ ↪`|"justify"|…`	Riguarda i componenti a blocchi e serve per dichiarare l'allineamento da usare. Al posto di indicare un allineamento a destra o a sinistra, si preferiscono definizioni riferite alla direzione di scrittura.
`text-indent="`***dimensione***`"|"`***n***`%"`	Riguarda i componenti a blocchi e serve per dichiarare il rientro della prima riga. Un valore percentuale si riferisce all'ampiezza del blocco che lo contiene.

Le sezioni successive non esauriscono l'argomento trattato, perché diversi elementi e molti attributi non vengono descritti affatto. Per ottenere le informazioni mancanti e in modo più dettagliato, si consulti la documentazione annotata nella bibliografia che appare alla fine del capitolo.

52.5.1 Blocchi di testo comuni

Un blocco di testo comparabile all'elemento '**`P`**' di HTML, si ottiene con l'elemento '**`fo:block`**', che però può contenere anche altri blocchi, oltre al testo lineare puro e semplice:

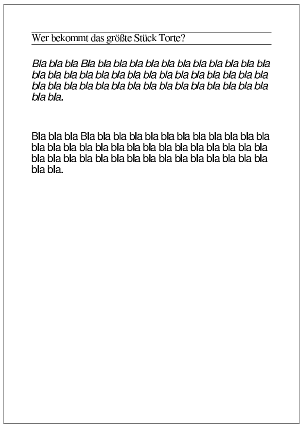
```
<fo:block
        [attributi]>
    …
</fo:block>
```

L'esempio seguente mostra l'uso di alcuni attributi per modificare il linguaggio, per inserire dei bordi, per modificare il carattere e per inserire dei margini:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE fo:root SYSTEM "fo.dtd">
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
    <fo:layout-master-set>
        <fo:simple-page-master master-name="normale"
            page-width="210mm" page-height="297mm"
            margin-top="2cm"   margin-bottom="2cm"
            margin-left="2cm"  margin-right="2cm">
            <fo:region-body />
        </fo:simple-page-master>
    </fo:layout-master-set>
    <fo:page-sequence master-reference="normale"
        language="it" country="it">
        <fo:flow flow-name="xsl-region-body">
            <fo:block language="de"       country="de"
                      font-family="serif" font-size="7mm"
                      border-before-style="solid"
                      border-after-style="solid">
                Wer bekommt das gr&#x00F6;&#x00DF;te
                St&#x00FC;ck Torte?
            </fo:block>
            <fo:block font-style="italic" font-size="7mm"
                      space-before="1cm"  space-after="1cm">
                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
```

```
                bla bla bla.
            </fo:block>
            <fo:block font-size="7mm"
                      space-before="1cm" space-after="1cm">
                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
                bla bla bla.
            </fo:block>
        </fo:flow>
    </fo:page-sequence>
</fo:root>
```

Il risultato della composizione si può osservare nelle figura seguente:


Wer bekommt das größte Stück Torte?

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 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 bla bla bla bla bla bla.

Dal momento che tutti gli elementi '**`fo:block`**' dell'esempio utilizzano un carattere di 7 mm, si può sfruttare l'ereditarietà della proprietà, inserendo tutti questi blocchi in un elemento '**`fo:block`**' complessivo (riga 15) in cui si dichiara una volta sola tale dimensione:

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <!DOCTYPE fo:root SYSTEM "fo.dtd">
3  <fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
4      <fo:layout-master-set>
5          <fo:simple-page-master master-name="normale"
6              page-width="210mm" page-height="297mm"
7              margin-top="2cm" margin-bottom="2cm"
8              margin-left="2cm" margin-right="2cm">
9              <fo:region-body />
10         </fo:simple-page-master>
11     </fo:layout-master-set>
12     <fo:page-sequence  master-reference="normale"
13         language="it" country="it">
14         <fo:flow flow-name="xsl-region-body">
15             <fo:block font-size="7mm">
16                 <fo:block language="de" country="de"
17                     font-family="serif"
18                     font-size="7mm"
19                     border-before-style="solid"
20                     border-after-style="solid">
21                     Wer bekommt das
22                     gr&#x00F6;&#x00DF;te
23                     St&#x00FC;ck Torte?
24                 </fo:block>
25                 <fo:block font-style="italic"
26                     space-before="1cm"
27                     space-after="1cm">
28                     Bla bla bla Bla bla bla bla bla bla
29                     bla bla bla bla bla bla bla bla bla
30                     bla bla bla bla bla bla bla bla bla
31                     bla bla bla bla bla bla bla bla bla
32                     bla bla bla bla bla bla bla bla bla
33                     bla bla.
34                 </fo:block>
35                 <fo:block space-before="1cm"
36                     space-after="1cm">
37                     Bla bla bla Bla bla bla bla bla bla
38                     bla bla bla bla bla bla bla bla bla
39                     bla bla bla bla bla bla bla bla bla
40                     bla bla bla bla bla bla bla bla bla
41                     bla bla bla bla bla bla bla bla bla
42                     bla bla.
43                 </fo:block>
44             </fo:block>
45         </fo:flow>
46     </fo:page-sequence>
47 </fo:root>
```


Il risultato che si ottiene è lo stesso già visto nella figura precedente.

52.5.2 Testo lineare

In un contesto lineare, si può inserire testo o elementi che non creano un blocco. In modo particolare, l'elemento '**`fo:inline`**', con l'uso dei suoi attributi, consente di modificare le caratteristiche del testo, come il carattere, lo spostamento sulla linea di riferimento (apici e pedici) e il colore:

```
<fo:inline
    [font-family="nome"|"serif"|"sans-serif"
        |"cursive"|"fantasy"|"monospace"]
    [font-size="altezza"|"xx-small"|"x-small"
        |"small"|"medium"|"large"|"x-large"
        |"xx-large"|"larger"|"smaller"
        |"n%"|"nem"]
    [font-style="normal"|"italic"|"oblique"
        |"backslant"]
    [font-stretch="ultra-condensed"|"extra-condensed"
        |"condensed"|"semi-condensed"
        |"normal"|"semi-expanded"
        |"expanded"|"extra-condensed"
        |"ultra-expanded"]
    [font-variant="normal"|"small-caps"]
    [font-weight="normal"|"bold"|"bolder"|"lighter"]
    [baseline-shift="sub"|"super"|...]
    [color="colore"]
    [altri_attributi]>
    [contenuto_lineare]
</fo:inline>
```

L'esempio seguente mostra solo un estratto di un file XSL-FO, con un blocco che contiene testo lineare, all'interno del quale appaiono alcuni elementi '**`fo:inline`**' con lo scopo di modificare il carattere, il colore e lo spostamento verticale:

```
<fo:block>Bla bla bla <fo:inline font-style="italic">testo
in corsivo</fo:inline> bla bla bla <fo:inline
font-weight="bold">H<fo:inline
baseline-shift="sub">2</fo:inline>O</fo:inline> bla bla bla
<fo:inline font-family="monospace">ls -l | sort &gt;
prova</fo:inline> bla bla bla <fo:inline color="blue">testo
in blu</fo:inline> bla bla bla <fo:inline
font-variant="small-caps">testo in maiuscoletto</fo:inline>
bla bla bla.</fo:block>
```

Nell'ambito di un contesto lineare, eccezionalmente, è possibile inserire un blocco, attraverso l'elemento '**`fo:inline-container`**':

```
<fo:inline-container
        [width="ampiezza"]
        [altri_attributi]>
    [contenuto_blocco]
</fo:inline-container>
```

52.5.3 Elenchi

Gli elenchi sono definiti attraverso l'elemento '**`fo:list-block`**', che è, evidentemente, un blocco:

```
<fo:list-block [attributi]>
    <fo:list-item [attributi]>
        <fo:list-item-label [attributi]>
            blocchi
        </fo:list-item-label>
        <fo:list-item-body [attributi]>
            blocchi
        </fo:list-item-body>
    <fo:/list-item>
    …
</fo:list-block>
```

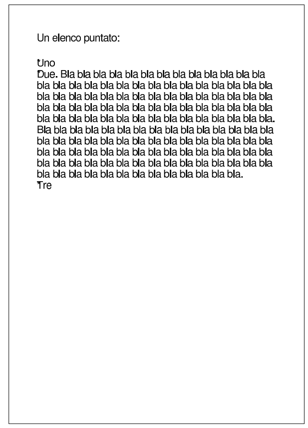
In pratica, l'elemento '**`fo:list-block`**' contiene obbligatoriamente elementi '**`fo:list-item`**', i quali, a loro volta, contengono la coppia '**`fo:list-item-label`**' e '**`fo:list-item-body`**'. L'esempio seguente costruisce un elenco puntato molto semplice, dove i lembi che segnalano l'inizio di ogni punto sono costituiti da un asterisco:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE fo:root SYSTEM "fo.dtd">
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
    <fo:layout-master-set>
        <fo:simple-page-master     master-name="normale"
                page-width="210mm" page-height="297mm"
                margin-top="2cm"   margin-bottom="2cm"
                margin-left="2cm"  margin-right="2cm">
            <fo:region-body />
        </fo:simple-page-master>
    </fo:layout-master-set>
    <fo:page-sequence master-reference="normale"
        language="it" country="it">
        <fo:flow flow-name="xsl-region-body">
            <fo:block font-size="7mm">
                <fo:block space-after="1cm">
                    Un elenco puntato:
                </fo:block>
                <fo:list-block  space-after="1cm">
                    <fo:list-item>
                        <fo:list-item-label>
                            <fo:block>*</fo:block>
                        </fo:list-item-label>
                        <fo:list-item-body>
                            <fo:block>Uno</fo:block>
                        </fo:list-item-body>
                    </fo:list-item>
                    <fo:list-item>
                        <fo:list-item-label>
                            <fo:block>*</fo:block>
                        </fo:list-item-label>
                        <fo:list-item-body>
                            <fo:block>
                                Due. 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 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.
                            </fo:block>
                            <fo:block>
                                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 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.
                            </fo:block>
                        </fo:list-item-body>
```



```
                    </fo:list-item>
                    <fo:list-item>
                        <fo:list-item-label>
                            <fo:block>*</fo:block>
                        </fo:list-item-label>
                        <fo:list-item-body>
                            <fo:block>Tre</fo:block>
                        </fo:list-item-body>
                    </fo:list-item>
                </fo:list-block>
            </fo:block>
        </fo:flow>
    </fo:page-sequence>
</fo:root>
```

Tuttavia, se si esegue la composizione si può osservare che gli asterischi si trovano a essere sovrapposti al corpo delle voci:

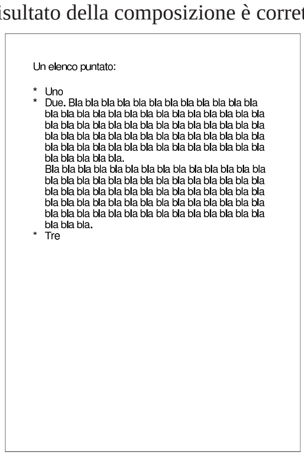


Per evitare che ciò accada, è necessario intervenire con dei rientri appropriati, utilizzando gli attributi dei blocchi che sono contenuti, sia come etichette, sia come corpo delle voci elencate. Per la precisione, si usa l'attributo '**`end-indent`**' nei blocchi delle etichette e l'attributo '**`start-indent`**' nei blocchi del corpo. Per fare in modo che queste distanze vengano calcolate automaticamente, si usano delle funzioni interne a XSL-FO, in grado di restituire il valore appropriato al contesto. Si tratta rispettivamente di ***label-end()*** e di ***body-start()***. Pertanto, l'esempio va modificato come segue:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE fo:root SYSTEM "fo.dtd">
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
    <fo:layout-master-set>
        <fo:simple-page-master master-name="normale"
            page-width="210mm" page-height="297mm"
            margin-top="2cm" margin-bottom="2cm"
            margin-left="2cm" margin-right="2cm">
            <fo:region-body />
        </fo:simple-page-master>
    </fo:layout-master-set>
    <fo:page-sequence master-reference="normale"
        language="it" country="it">
        <fo:flow flow-name="xsl-region-body">
            <fo:block font-size="7mm">
                <fo:block space-after="1cm">
                    Un elenco puntato:
                </fo:block>
                <fo:list-block space-after="1cm">
                    <fo:list-item>
                        <fo:list-item-label>
                            <fo:block
                                end-indent="label-end()">
                                *
                            </fo:block>
                        </fo:list-item-label>
                        <fo:list-item-body>
                            <fo:block
                                start-indent="body-start()">
                                Uno
                            </fo:block>
                        </fo:list-item-body>
                    </fo:list-item>
```

```
                    <fo:list-item>
                        <fo:list-item-label>
                            <fo:block
                                end-indent="label-end()">
                                *
                            </fo:block>
                        </fo:list-item-label>
                        <fo:list-item-body>
                            <fo:block
                                start-indent="body-start()">
                                Due. 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 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.
                            </fo:block>
                            <fo:block
                                start-indent="body-start()">
                                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 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.
                            </fo:block>
                        </fo:list-item-body>
                    </fo:list-item>
                    <fo:list-item>
                        <fo:list-item-label>
                            <fo:block
                                end-indent="label-end()">
                                *
                            </fo:block>
                        </fo:list-item-label>
                        <fo:list-item-body>
                            <fo:block
                                start-indent="body-start()">
                                Tre
                            </fo:block>
                        </fo:list-item-body>
                    </fo:list-item>
                </fo:list-block>
            </fo:block>
        </fo:flow>
    </fo:page-sequence>
</fo:root>
```

Questa volta il risultato della composizione è corretto:



52.5.4 Tabelle

Come si può immaginare, la realizzazione di una tabella con XSL-FO è un'operazione abbastanza complessa, anche nelle situazioni più semplici:

```
<fo:table-and-caption [attributi]>
    [<fo:table-caption [attributi]>
        blocchi
    </fo:table-caption>]
    <fo:table
            [table-layout="auto"…"fixed"]
            [altri_attributi]>
        [<fo:table-column
                [table-column="ampiezza"]
                [attributi] />]
        …
        [<fo:table-header [attributi]>
            righe
        </fo:table-header>]
        [<fo:table-footer [attributi]>
            righe
        </fo:table-footer>]
        <fo:table-body [attributi]>
            righe
        </fo:table-body>
    </fo:table-caption>
</fo:table-and-caption>
```

Il significato dei vari elementi che appaiono nel modello sintattico dovrebbe essere intuitivo; si osservi in particolare che una tabella si può realizzare anche partendo soltanto dall'elemento '**`fo:table`**', quando non è importante l'inserimento di una didascalia con l'elemento '**`fo:table-caption`**'. Le righe interne all'intestazione, al piè di pagina e al corpo, vengono inserite nel modo seguente:

```
<fo:table-row [attributi]>
    <fo:table-cell [attributi]>
        blocchi
    </fo:table-cell>
</fo:table-row>
```

L'ampiezza delle colonne, i bordi e altre caratteristiche della tabella vengono gestiti attraverso gli attributi. Teoricamente, l'elemento '**`fo:table`**' dispone dell'attributo '**`table-layout`**', con il quale si potrebbe richiedere un adattamento automatico della larghezza delle colonne; in pratica questa funzionalità è abbastanza difficile da trovare, pertanto si è costretti a usare l'attributo '**`column-width`**' degli elementi '**`fo:table-column`**'.

L'esempio seguente mostra una tabella molto semplice e anche molto povera, perché manca qualunque bordo e perché viene definita senza l'elemento '**`fo:table-and-caption`**', in modo da garantire il funzionamento con le prime edizioni di FOP:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE fo:root SYSTEM "fo.dtd">
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
    <fo:layout-master-set>
        <fo:simple-page-master master-name="normale"
            page-width="210mm" page-height="297mm"
            margin-top="2cm"   margin-bottom="2cm"
            margin-left="2cm"  margin-right="2cm">
            <fo:region-body />
        </fo:simple-page-master>
    </fo:layout-master-set>
    <fo:page-sequence master-reference="normale"
        language="it" country="it">
        <fo:flow flow-name="xsl-region-body">
            <fo:block font-size="7mm">
                <fo:block>Tabella di esempio. 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 bla bla bla
                bla bla bla bla bla bla bla bla bla
                bla</fo:block>
                <fo:table table-layout="fixed">
                    <fo:table-column column-width="5cm" />
                    <fo:table-column column-width="7cm" />
                    <fo:table-column column-width="5cm" />
                    <fo:table-header>
                        <fo:table-row>
                            <fo:table-cell>
                                <fo:block>
                                    Parola di controllo
                                </fo:block>
                            </fo:table-cell>
                            <fo:table-cell>
                                <fo:block>
                                    Competenza
                                </fo:block>
                            </fo:table-cell>
                            <fo:table-cell>
                                <fo:block>
                                    Condizione o valore
                                    predefinito
                                </fo:block>
                            </fo:table-cell>
                        </fo:table-row>
                    </fo:table-header>
                    <fo:table-body>
                        <fo:table-row>
                            <fo:table-cell>
                                <fo:block>
                                    \hoffset
                                </fo:block>
                            </fo:table-cell>
                            <fo:table-cell>
                                <fo:block>
                                    Posizione iniziale dei
                                    paragrafi nella pagina.
                                </fo:block>
                            </fo:table-cell>
                            <fo:table-cell>
                                <fo:block>
                                    0
                                </fo:block>
                            </fo:table-cell>
                        </fo:table-row>
                        <fo:table-row>
                            <fo:table-cell>
                                <fo:block>
                                    \hsize
                                </fo:block>
                            </fo:table-cell>
                            <fo:table-cell>
                                <fo:block>
                                    Larghezza del paragrafo
                                    a partire da \hoffset.
                                </fo:block>
                            </fo:table-cell>
                            <fo:table-cell>
                                <fo:block>
                                    6,5 pollici
                                </fo:block>
                            </fo:table-cell>
                        </fo:table-row>
                        <fo:table-row>
                            <fo:table-cell>
                                <fo:block>
                                    \parindent
                                </fo:block>
                            </fo:table-cell>
                            <fo:table-cell>
                                <fo:block>
                                    Rientro della prima
                                    riga.
                                </fo:block>
                            </fo:table-cell>
                            <fo:table-cell>
                                <fo:block>
                                    20 punti
                                </fo:block>
                            </fo:table-cell>
                        </fo:table-row>
                        <fo:table-row>
                            <fo:table-cell>
                                <fo:block>
                                    \baselineskip
                                </fo:block>
                            </fo:table-cell>
                            <fo:table-cell>
                                <fo:block>
                                    Distanza tra la base di
                                    una riga e la base della
                                    riga successiva.
                                </fo:block>
                            </fo:table-cell>
                            <fo:table-cell>
                                <fo:block>
                                    12 punti
                                </fo:block>
                            </fo:table-cell>
                        </fo:table-row>
                        <fo:table-row>
                            <fo:table-cell>
                                <fo:block>
                                    \parskip
                                </fo:block>
                            </fo:table-cell>
                            <fo:table-cell>
                                <fo:block>
                                    Distanza aggiuntiva tra
                                    i paragrafi.
                                </fo:block>
                            </fo:table-cell>
                            <fo:table-cell>
                                <fo:block>
                                    0
                                </fo:block>
                            </fo:table-cell>
                        </fo:table-row>
                        <fo:table-row>
                            <fo:table-cell>
                                <fo:block>
                                    \raggedright
                                </fo:block>
                            </fo:table-cell>
                            <fo:table-cell>
                                <fo:block>
                                    Allinea il testo a
                                    sinistra.
                                </fo:block>
                            </fo:table-cell>
                            <fo:table-cell>
                                <fo:block>
                                    allineato
                                    simultaneamente a
                                    sinistra e a destra
                                </fo:block>
                            </fo:table-cell>
                        </fo:table-row>
                    </fo:table-body>
                </fo:table>
            </fo:block>
        </fo:flow>
    </fo:page-sequence>
</fo:root>
```

Ecco il risultato:

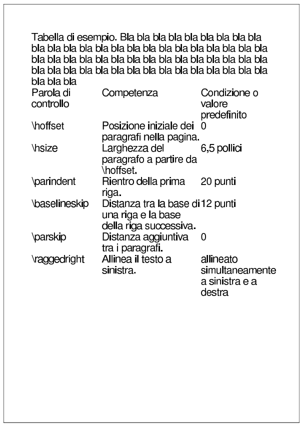


Tabella di esempio. Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla

Parola di controllo	Competenza	Condizione o valore predefinito
\hoffset	Posizione iniziale dei paragrafi nella pagina.	0
\hsize	Larghezza del paragrafo a partire da \hoffset.	6,5 pollici
\parindent	Rientro della prima riga.	20 punti
\baselineskip	Distanza tra la base di una riga e la base della riga successiva.	12 punti
\parskip	Distanza aggiuntiva tra i paragrafi.	0
\raggedright	Allinea il testo a sinistra.	allineato simultaneamente a sinistra e a destra

Per introdurre dei bordi, almeno teoricamente, si può intervenire dal blocco contenuto nella singola cella, fino all'elemento '**`fo:table`**' più esterno; tuttavia, a seconda del programma di elaborazione che si utilizza, può darsi che in alcuni punti i bordi non vengano presi in considerazione.

52.5.5 Inserzione di immagini

L'inserzione di un'immagine si ottiene con l'elemento '**`fo:external-graphic`**', che è vuoto e appartiene a un contesto lineare:

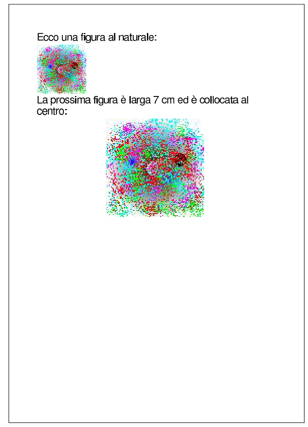
```
<fo:external-graphic>
        src="file"|"indirizzo_uri"
        [scaling="uniform"|"non-uniform"]
        [height="auto"|"altezza"]
        [width="auto"|"larghezza"]
        [altri_attributi] />
```

Se non si indicano delle dimensioni attraverso gli attributi '**`height`**' o '**`width`**', l'immagine viene inserita alla sua dimensione naturale. Se si modifica una sola dimensione, normalmente si ottiene l'adattamento dell'altro valore in modo relativo; tuttavia, attraverso l'opzione '**`scaling="non-uniform"`**', è possibile deformare l'immagine.

Il file o l'indirizzo URI dell'immagine deve fare riferimento a un formato compatibile con le capacità elaborative del sistema usato per la composizione. Di solito vanno bene almeno i formati GIF e JPG.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE fo:root SYSTEM "fo.dtd">
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
    <fo:layout-master-set>
        <fo:simple-page-master master-name="normale"
            page-width="210mm" page-height="297mm"
            margin-top="2cm"   margin-bottom="2cm"
            margin-left="2cm"  margin-right="2cm">
            <fo:region-body />
        </fo:simple-page-master>
    </fo:layout-master-set>
    <fo:page-sequence master-reference="normale"
        language="it" country="it">
        <fo:flow flow-name="xsl-region-body">
            <fo:block font-size="7mm">
                <fo:block>
                    Ecco una figura al naturale:
                </fo:block>
                <fo:block>
                    <fo:external-graphic
                            src="esempio.jpg" />
                </fo:block>
                <fo:block>
                    La prossima figura &#x00E8; larga 7 cm
                    ed &#x00E8; collocata al centro:
                </fo:block>
                <fo:block   text-align="center">
                    <fo:external-graphic
                            src="esempio.jpg"
                            scaling="uniform"
                            width="7cm" />
                </fo:block>
            </fo:block>
        </fo:flow>
    </fo:page-sequence>
</fo:root>
```


L'esempio mostra l'inserzione della stessa immagine due volte, in modi differenti. In particolare, la seconda volta l'immagine viene centrata orizzontalmente per mezzo dell'attributo '**`text-align`**' dell'elemento '**`fo:block`**' che la contiene.



Ecco una figura al naturale:

La prossima figura è larga 7 cm ed è collocata al centro:

52.5.6 Linee orizzontali

Si può realizzare facilmente una linea orizzontale con un elemento '**`fo:block`**' bordato, magari anche vuoto:

```
<fo:block
        border-after-style="stile"
        [border-after-width="spessore"]
        [border-after-color="colore"]
        [altri_attributi] />
```

Per inserire delle linee orizzontali in un contesto lineare, ovvero all'interno del testo, si usa l'elemento '**`fo:leader`**':

```
<fo:leader
        leader-pattern="use-content"
        [leader-length="lunghezza"]
        [altri_attributi]>
    modello
</fo:leader>
```

```
<fo:leader
        leader-pattern="space"|"rule"|"dots"
        [leader-length="lunghezza"]
        [rule-style="stile"]
        [rule-thickness="spessore"]
        [altri_attributi] />
```

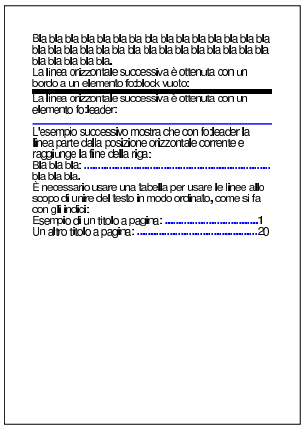
Se l'elemento contiene qualcosa e all'attributo '**`leader-pattern`**' viene associato il valore '**`use-content`**', questo viene usato per costruire la linea (per esempio si potrebbe realizzare una linea fatta di asterischi o di sequenze più complesse). Tuttavia, di solito si vogliono realizzare soltanto linee continue, spezzate o puntate, pertanto l'elemento si usa prevalentemente vuoto.

È importante osservare che la linea che si ottiene raggiunge la fine dello spazio orizzontale disponibile, pertanto, per realizzare linee di collegamento come quelle degli indici, dove appare un titolo alla sinistra e un numero di pagina alla destra, occorre inserire tutto in una tabella. Nell'esempio seguente vengono mostrati diversi ca-

si, compresa la simulazione di un indice elementare, inserito in una tabella:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE fo:root SYSTEM "fo.dtd">
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
    <fo:layout-master-set>
        <fo:simple-page-master master-name="normale"
            page-width="210mm" page-height="297mm"
            margin-top="2cm"   margin-bottom="2cm"
            margin-left="2cm"  margin-right="2cm">
            <fo:region-body />
        </fo:simple-page-master>
    </fo:layout-master-set>
    <fo:page-sequence master-reference="normale"
        language="it" country="it">
        <fo:flow flow-name="xsl-region-body">
            <fo:block font-size="7mm">
                <fo:block>
                    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.
                </fo:block>
                <fo:block>
                    La linea orizzontale successiva &#x00E8;
                    ottenuta con un bordo a un elemento
                    fo:block vuoto:
                </fo:block>
                <fo:block
                        border-after-style="solid"
                        border-after-width="3mm"
                        border-after-color="black" />
                <fo:block>
                    La linea orizzontale successiva &#x00E8;
                    ottenuta con un elemento fo:leader:
                </fo:block>
                <fo:block>
                    <fo:leader
                            leader-pattern="rule"
                            rule-style="solid"
                            rule-thickness="1mm"
                            color="blue"/>
                </fo:block>
                <fo:block>
                    L'esempio successivo mostra che con
                    fo:leader la linea parte dalla posizione
                    orizzontale corrente e raggiunge la fine
                    della riga:
                </fo:block>
                <fo:block>
                    Bla bla bla:
                    <fo:leader
                        leader-alignment="reference-area"
                        leader-pattern="dots"
                        rule-style="solid"
                        rule-thickness="1mm"
                        color="blue"/>
                    bla bla bla.
                </fo:block>
                <fo:block>
                    &#x00C8; necessario usare una tabella
                    per usare le linee allo scopo di unire
                    del testo in modo ordinato, come si fa
                    con gli indici:
                </fo:block>
                <fo:table   table-layout="fixed">
                    <fo:table-column column-width="16cm"/>
                    <fo:table-column column-width="1cm"/>
                    <fo:table-body>
                        <fo:table-row>
                            <fo:table-cell>
                                <fo:block>
                                    Esempio di un titolo a
                                    pagina:
                                    <fo:leader
                                      leader-pattern="dots"
                                      rule-style="solid"
                                      rule-thickness="1mm"
                                      color="blue"/>
                                </fo:block>
                            </fo:table-cell>
                            <fo:table-cell>
                                <fo:block>
                                    1
                                </fo:block>
                            </fo:table-cell>
                        </fo:table-row>
                        <fo:table-row>
                            <fo:table-cell>
                                <fo:block>
                                    Un altro titolo a
                                    pagina:
                                    <fo:leader
                                       leader-pattern="dots"
                                       rule-style="solid"
                                       rule-thickness="1mm"
                                       color="blue"/>
                                </fo:block>
                            </fo:table-cell>
                            <fo:table-cell>
                                <fo:block>
                                    20
                                </fo:block>
                            </fo:table-cell>
                        </fo:table-row>
                    </fo:table-body>
                </fo:table>
            </fo:block>
        </fo:flow>
    </fo:page-sequence>
</fo:root>
```

Ecco il risultato della composizione dell'esempio:



52.5.7 Riferimenti incrociati

XSL-FO ha una gestione molto sofisticata dei riferimenti ipertestuali; tuttavia, qui si vuole mostrare solo l'uso più semplice. In generale, la maggior parte degli elementi può contenere l'attributo '**id**', con lo scopo di associare una stringa di identificazione univoca, in modo da consentire successivamente di raggiungere il contenuto di questi elementi attraverso un riferimento interno.

Per fare riferimento a qualcosa, si usa l'elemento '**fo:basic-link**' (nell'ambito di un contesto lineare), specificando se si tratta di un riferimento interno o esterno, con l'uso alternativo degli attributi '**internal-destination**' e '**external-destination**':

```
<fo:basic-link
        {internal-destination="stringa_id"}↩
↪       |{external-destination="indirizzo_uri"}
        [altri_attributi]>
    contenuto
</fo:basic-link>
```

Il testo contenuto nell'elemento serve per individuare la zona su cui funziona la selezione del riferimento, come nell'esempio seguente, dove la parola «qui» serve come superficie su cui fare un clic con il mouse:

```
Per raggiungere il sito, basta fare un clic <fo:basic-link
external-destination="http://www.brot.dg">qui</fo:basic-link>.
```

Per fare riferimento a una pagina, in modo da ottenerne il numero nella composizione stampata, si usano due elementi per due situazioni differenti:

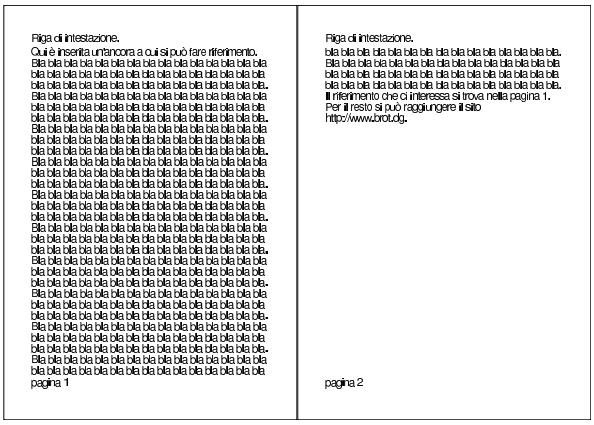
```
<fo:page-number [attributi] />
```

```
<fo:page-number-citation
        ref-id="stringa_a_cui_si_fa_riferimento"
        [attributi] />
```

Nel primo caso, si ottiene il numero della pagina corrente; nel secondo, si ottiene il numero di pagina a cui si fa riferimento con la stringa indicata con l'attributo '**ref-id**'. L'esempio seguente riassume l'uso di questi elementi, dove in particolare si inserisce il numero di pagina nella regione inferiore:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE fo:root SYSTEM "fo.dtd">
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
    <fo:layout-master-set>
        <fo:simple-page-master
                master-name="normale"
                page-width="210mm"
                page-height="297mm"
                margin-top="2cm"
                margin-bottom="2cm"
                margin-left="2cm"
                margin-right="2cm">
            <fo:region-body margin-top="1cm"
                margin-bottom="1cm"/>
            <fo:region-before extent="1cm" />
            <fo:region-after  extent="1cm" />
        </fo:simple-page-master>
    </fo:layout-master-set>
    <fo:page-sequence
            master-reference="normale"
            initial-page-number="1"
            force-page-count="even"
            language="it"
            country="it">
        <fo:static-content flow-name="xsl-region-before">
            <fo:block font-size="7mm">
                Riga di intestazione.
            </fo:block>
        </fo:static-content>
        <fo:static-content flow-name="xsl-region-after">
            <fo:block font-size="7mm">
                pagina <fo:page-number />
            </fo:block>
        </fo:static-content>
        <fo:flow flow-name="xsl-region-body">
            <fo:block font-size="7mm">
                <fo:block id="riferimento">
                    Qui &#x00E8; inserita un'ancora a cui si
                    pu&#x00F2; fare riferimento.
                </fo:block>
                <fo:block>
                    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
                    bla bla bla bla bla.
                </fo:block>
                <!-- Inserire qui altri fo:block per
                     ottenere piu' pagine -->
                <fo:block>
                    Il riferimento che ci interessa si trova
                    nella <fo:basic-link
                    internal-destination="riferimento">pagina
                    <fo:page-number-citation
                    ref-id="riferimento"/></fo:basic-link>.
                    Per il resto <fo:basic-link
                external-destination="http://www.brot.dg">si
                    pu&#x00F2; raggiungere il sito
                    http://www.brot.dg</fo:basic-link>.
                </fo:block>
            </fo:block>
        </fo:flow>
    </fo:page-sequence>
</fo:root>
```


Ecco come potrebbe apparire il risultato della composizione:



52.5.8 Componenti autonomi rispetto al testo

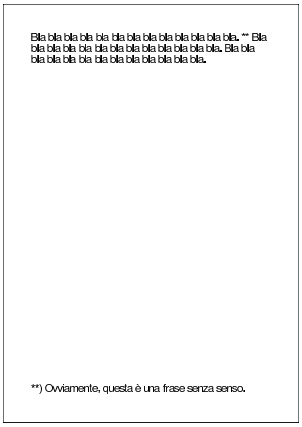
Esistono due tipi di componenti che, nella composizione, non si collocano nella posizione in cui appaiono nel corso del documento. Si tratta di note a piè pagina e di oggetti fluttuanti. Le note a piè pagina si inseriscono nell'ambito di un contesto lineare e si ottengono con questa sintassi:

```
<fo:footnote>
    <fo:inline [attributi]>
        simbolo
    </fo:inline>
    <fo:footnote-body>
        blocchi
    </fo:footnote-body>
</fo:footnote>
```

In pratica, l'elemento '**fo:footnote**' contiene necessariamente un elemento '**fo:inline**', il cui scopo è quello di inserire un simbolo o qualunque cosa serva per rinviare il lettore alla nota a piè pagina; subito dopo, l'elemento '**fo:footnote-body**' contiene uno o più blocchi che contengono la nota vera e propria. Si tenga in considerazione il fatto che, probabilmente, si vuole inserire all'inizio delle note lo stesso simbolo usato per il richiamo.

```
<fo:block>
    Bla bla bla bla bla bla bla bla bla bla bla bla bla.
    <fo:footnote>
        <fo:inline>**</fo:inline>
        <fo:footnote-body>
            <fo:block>
                **) Ovviamente, questa &#x00E8; una
                frase senza senso.
            </fo:block>
        </fo:footnote-body>
    </fo:footnote>
    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.
</fo:block>
```

L'esempio mostra solo una porzione di un file XSL-FO, che, nella composizione finale potrebbe apparire in modo simile a questo:



Gli oggetti fluttuanti si inseriscono all'interno dell'elemento '**`fo:float`**', che si comporta come un blocco:

```
<fo:float
        float="before"|"start"|"end"|…
        [clear="posizione"]>
    blocchi
</fo:float>
```

In pratica, con l'aiuto dell'attributo '**`float`**' si può stabilire in che modo debba essere collocato l'oggetto fluttuante, mentre l'attributo '**`clear`**' serve a definire un comportamento in prossimità di altri oggetti fluttuanti.

Non vengono mostrati esempi sull'uso di questo elemento, anche se si può intuire la sua importanza in presenza di immagini e piccole tabelle che è bene collocare in modo dinamico nel corpo di un documento.

52.6 XSLT e XSL-FO

Fino a questo punto, XSLT e XSL-FO sono stati descritti separatamente, per evitare confusione, mentre fanno parte di un sistema integrato. Una volta compreso il meccanismo di XSLT e di XSL-FO, non dovrebbe essere difficile unire le due cose per trasformare un documento XML in un file XSL-FO, allo scopo di arrivare poi alla composizione finale.

In precedenza è stato mostrato un file XML molto semplice, associando a questo la trasformazione in formati comuni:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE relazione [
    <!ELEMENT relazione     (titolo?, data, contenuto)>
    <!ATTLIST relazione
        xml:lang   CDATA    "">
    <!ELEMENT titolo        (#PCDATA)>
    <!ELEMENT data          (#PCDATA)>
    <!ELEMENT contenuto     (paragrafo+, firma+)>
    <!ELEMENT paragrafo     (#PCDATA)>
    <!ELEMENT firma         (#PCDATA)>
]>

<relazione xml:lang="it">
<titolo>Relazione introduttiva su XML</titolo>

<data>11/11/2011</data>

<contenuto>

<paragrafo>XML sta per Extensible Markup Language.    bla
bla bla... Perché,... così,... perciò,...
sarà...</paragrafo>

<paragrafo>Bla, bla, bla....</paragrafo>

<firma>Pinco Pallino</firma>

</contenuto>
</relazione>
```


Per trasformare questo tipo di file in un foglio di stile XSL-FO, si può usare un foglio di stile XSLT come quello seguente; si osservi anche la presenza della dichiarazione del dominio applicativo associato al prefisso '**`fo:`**'.

```
<xsl:stylesheet
    version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:fo="http://www.w3.org/1999/XSL/Format">
    <xsl:output
        method="xml"
        indent="yes"
        encoding="UTF-8" />
    <xsl:template match="/">
        <fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
            <fo:layout-master-set>
                <fo:simple-page-master master-name="normale"
                    page-width="210mm" page-height="297mm"
                    margin-top="2cm"   margin-bottom="2cm"
                    margin-left="2cm"  margin-right="2cm">
                    <fo:region-body />
                </fo:simple-page-master>
            </fo:layout-master-set>
            <fo:page-sequence master-reference="normale"
                language="{relazione/@xml:lang}">
                <fo:flow flow-name="xsl-region-body">
        <xsl:apply-templates />
                </fo:flow>
            </fo:page-sequence>
        </fo:root>
    </xsl:template>
    <xsl:template match="titolo">
        <fo:block font-size="4mm" space-after="1cm"
            text-align="right" color="blue">
            <xsl:apply-templates />
        </fo:block>
    </xsl:template>
    <xsl:template match="data">
        <fo:block space-after="5mm">
            <xsl:apply-templates />
        </fo:block>
    </xsl:template>
    <xsl:template match="paragrafo">
        <fo:block>
            <xsl:apply-templates />
        </fo:block>
    </xsl:template>
    <xsl:template match="firma">
        <fo:block>
            <xsl:apply-templates />
        </fo:block>
    </xsl:template>
</xsl:stylesheet>
```

In questo caso la trasformazione genera il file XSL-FO seguente:

```
<?xml version="1.0" encoding="UTF-8"?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
<fo:layout-master-set>
<fo:simple-page-master master-name="normale"
    page-width="210mm" page-height="297mm" margin-top="2cm"
    margin-bottom="2cm" margin-left="2cm"
    margin-right="2cm">
<fo:region-body/>
</fo:simple-page-master>
</fo:layout-master-set>
<fo:page-sequence master-reference="normale" language="it">
<fo:flow flow-name="xsl-region-body">
<fo:block font-size="4mm" space-after="1cm"
    text-align="right"
    color="blue">Relazione introduttiva su XML</fo:block>

<fo:block space-after="5mm">11/11/2011</fo:block>



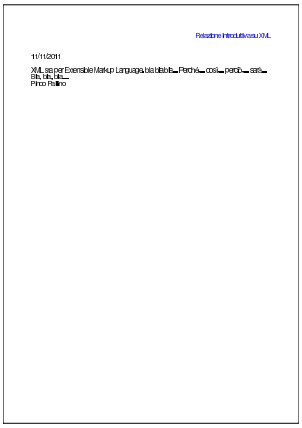
<fo:block>XML sta per Extensible Markup Language.    bla
bla bla... Perché,... così,... perciò,...
sarà...</fo:block>

<fo:block>Bla, bla, bla....</fo:block>
```

```
<fo:block>Pinco Pallino</fo:block>


</fo:flow>
</fo:page-sequence>
</fo:root>
```

Il risultato della composizione:



Per il resto, si veda quanto già descritto in precedenza a proposito di XSLT nella sezione 52.2.

52.7 XMLTeX e PassiveTeX

XMLTeX[4] è un sistema per comporre file XML attraverso TeX; PassiveTeX[5] è un insieme di fogli di stile aggiuntivi che, in particolare, consentono a XMLTeX di elaborare file XSL-FO.

52.7.1 XMLTeX

XMLTeX è un sistema che si affianca normalmente a LaTeX per trasformare un file XML in un documento pronto per la stampa, attraverso un metodo di trasformazione degli elementi e degli attributi del file di origine in comandi di LaTeX. Per fare questo occorre predisporre un «file di configurazione» nel quale si dichiarano queste sostituzioni. Viene proposto il solito esempio di file XML molto semplice, dove in particolare l'attributo '**`linguaggio`**' contiene un nome valido per LaTeX:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE relazione [
    <!ELEMENT relazione     (titolo?, data, contenuto)>
    <!ATTLIST relazione
        linguaggio  CDATA   "">
    <!ELEMENT titolo        (#PCDATA)>
    <!ELEMENT data          (#PCDATA)>
    <!ELEMENT contenuto     (paragrafo+, firma+)>
    <!ELEMENT paragrafo     (#PCDATA)>
    <!ELEMENT firma         (#PCDATA)>
]>

<relazione linguaggio="italian">
<titolo>Relazione introduttiva su XML</titolo>

<data>11/11/2011</data>

<contenuto>

<paragrafo>XML sta per Extensible Markup Language.    bla
bla bla... Perché,... così,... perciò,...
sarà...</paragrafo>

<paragrafo>Bla, bla, bla...</paragrafo>

<paragrafo>... ma soprattutto vorrei sorvolare sopra
questioni così complesse e arzigogolate come qualcuno
invece vorrebbe...</paragrafo>

<firma>Pinco Pallino</firma>
```



```
</contenuto>
</relazione>
```

Supponendo di dare a questo file il nome '`relazione.xml`', si può realizzare il file '`relazione.cfg`', con l'indicazione delle trasformazioni da apportare:

```
\XMLelement{relazione}
    {\XMLattribute{linguaggio}
        {\mylanguageattribute}{english}}{%
    \documentclass{article}
    \usepackage[OT2,OT1]{fontenc}
    \usepackage[\mylanguageattribute]{babel}
    \frenchspacing
    \begin{document}
}
{%
    \end{document}
}

\XMLelement{titolo}{}{%
    \xmlgrab
}
{
    \section{#1}
}

\XMLelement{data}{}{%
}
{
    \par
}

\XMLelement{contenuto}{}{%
}
{
}

\XMLelement{paragrafo}{}{%
}
{
    \par
}

\XMLelement{firma}{}{%
}
{
    \par
}
```

Come si vede si tratta di comandi TeX con cui si dichiarano gli abbinamenti tra elementi e attributi con comandi corrispondenti di LaTeX.

```
\XMLelement{nome_elemento}{[dichiarazione_attributi]}↩
↪{codice_iniziale}{codice_finale}
```

```
\XMLelement{nome_elemento}{[dichiarazione_attributi]}↩
↪{\xmlgrab}{codice_complessivo}
```

Il comando '**`\XMLelement`**' consente di dichiarare la trasformazione dei marcatori di un elemento in comandi LaTeX: gli ultimi due argomenti contengono rispettivamente il codice LaTeX di apertura e il codice di chiusura dell'elemento. Tuttavia, se nell'argomento destinato a ospitare il codice LaTeX di apertura si inserisce soltanto il comando '**`\xmlgrab`**', si può mettere tutto nell'argomento finale, usando la sigla '**`#1`**' per fare riferimento al contenuto dell'elemento.

Nel secondo argomento del comando '**`\XMLelement`**' si possono inserire comandi '**`\XMLattribute`**' per prelevare le informazioni sugli attributi:

```
\XMLattribute{nome_attributo}{nome_comando_tex}{[valore_predefinito]}
```

Osservando l'esempio del file '`relazione.cfg`' si può vedere che nella dichiarazione riferita all'elemento '**`relazione`**'

appare l'associazione dell'attributo '**linguaggio**' al comando '**\mylanguageattribute**', che poi viene usato successivamente. In pratica, il comando di LaTeX che nel file '`relazione.cfg`' appare come la riga seguente, è come se venisse sostituito con quella successiva:

```
    \usepackage[\mylanguageattribute]{babel}
```

```
    \usepackage[italian]{babel}
```

Nel file di configurazione, ovvero il file '`.cfg`', possono essere collocati altri comandi importanti; fortunatamente molte indicazioni essenziali fanno già parte della configurazione generale, che potrebbe trovarsi nel file '`/etc/texmf/xmltex/xmltex.cfg`'. Eventualmente si può approfondire la cosa leggendo la documentazione originale di XMLTeX.

Disponendo del file di configurazione per la trasformazione del documento XML, prima di passare alla composizione con LaTeX, si deve realizzare un piccolo file di collegamento, denominato, in questo caso, '`relazione.tex`':

```
\def\xmlfile{relazione.xml}
\input xmltex.tex
```

Si può passare così alla composizione di questo file '`relazione.tex`':

```
$ latex relazione.tex [invio]

This is TeX, Version 3.14159 (Web2C 7.4.5)
(./relazione.tex
LaTeX2e <2001/06/01>
Babel <v3.7h> and hyphenation patterns for american, french,
german, ngerman, danish, dutch, finnish, greek, italian,
latin, spanish, nohyphenation, loaded.
(/usr/share/texmf/tex/xmltex/base/xmltex.tex
xmltex version: 2002/06/25 v1.9 (Exp)

Encoding = utf-8
(/usr/share/texmf/tex/xmltex/config/xmltex.cfg)
(./relazione.cfg) (./relazione.xml
 Default: relazione linguaggio=""
 (/usr/share/texmf/tex/latex/base/article.cls
Document Class: article 2001/04/21 v1.4e Standard LaTeX
document class
(/usr/share/texmf/tex/latex/base/size10.clo))
(/usr/share/texmf/tex/latex/base/fontenc.sty
(/usr/share/texmf/tex/latex/cyrillic/ot2enc.def)
(/usr/share/texmf/tex/latex/base/ot1enc.def))
(/usr/share/texmf/tex/generic/babel/babel.sty
(/usr/share/texmf/tex/generic/babel/italian.ldf
(/usr/share/texmf/tex/generic/babel/babel.def)))
(./relazione.aux)
(/usr/share/texmf/tex/latex/cyrillic/ot2cmr.fd)
[1] (./relazione.aux) ) ) )
Output written on relazione.dvi (1 page, 692 bytes).
Transcript written on relazione.log.
```

Si ottiene così il file '`relazione.dvi`', da cui si può generare facilmente un file PostScript:

```
$ dvips -o relazione.ps relazione.dvi [invio]

This is dvips(k) 5.92b Copyright 2002 Radical Eye Software
(www.radicaleye.com) ' TeX output 2003.08.09:1533' ->
relazione.ps
<texc.pro><f7b6d320.enc><texps.pro>.
<cmr10.pfb><cmbx12.pfb>[1]
```

In modo analogo si può ottenere una composizione in formato PDF, utilizzando '**pdflatex**':

```
$ pdflatex relazione.tex [invio]
```

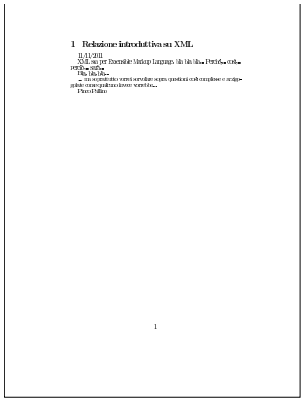
Eventualmente, entrambe le cose si possono fare in modo più semplice, senza bisogno di realizzare il file '`relazione.tex`', con i comandi seguenti, che servono a ottenere, rispettivamente, la composizione in formato DVI e PDF:

```
$ latex "&xmltex" relazione.tex [Invio]

$ pdflatex "&pdfxmltex" relazione.tex [Invio]
```


Nel caso dovessero essere presenti riferimenti incrociati, è necessario eseguire la composizione con LaTeX per tre volte di seguito (sempre con lo stesso comando).

Per concludere viene mostrato il risultato della composizione dell'esempio:



52.7.2 PassiveTeX

Come accennato all'inizio del capitolo, PassiveTeX è un insieme di fogli di stile da usare con XMLTeX per comporre direttamente file XSL-FO, attraverso LaTeX. Viene ripreso un esempio già apparso in un altro capitolo di file XSL-FO:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE fo:root SYSTEM "fo.dtd">
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
    <fo:layout-master-set>
        <fo:simple-page-master master-name="normale"
            page-width="210mm" page-height="297mm"
            margin-top="2cm"   margin-bottom="2cm"
            margin-left="2cm"  margin-right="2cm">
            <fo:region-body />
        </fo:simple-page-master>
    </fo:layout-master-set>
    <fo:page-sequence master-reference="normale"
        language="it" country="it">
        <fo:flow flow-name="xsl-region-body">
            <fo:block font-size="7mm">
                <fo:block>
                    Bla bla bla <fo:inline
                    font-style="italic">testo in
                    corsivo</fo:inline> bla bla bla
                    <fo:inline
                    font-weight="bold">H<fo:inline
            baseline-shift="sub">2</fo:inline>O</fo:inline>
                    bla bla bla <fo:inline
                    font-family="monospace">ls -l | sort
                    &gt; prova</fo:inline> bla bla bla
                    <fo:inline color="blue">testo in
                    blu</fo:inline> bla bla bla <fo:inline
                    font-variant="small-caps">testo in
                    maiuscoletto</fo:inline> bla bla bla.
                </fo:block>
            </fo:block>
        </fo:flow>
    </fo:page-sequence>
</fo:root>
```

In modo molto semplice, si arriva alla composizione di questo file con il comando seguente, supponendo che si tratti del file '`esempio-014.fo`':

```
$ latex "&xmltex" esempio-014.fo [Invio]
```

In questo modo si genera il file '`esempio-014.dvi`', dal quale si può ottenere un file PostScript nel modo consueto:

```
$ dvips -o esempio-014.ps esempio-014.dvi [Invio]
```

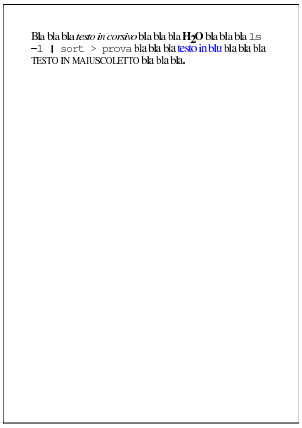
Ovviamente si può anche eseguire una composizione in formato

PDF direttamente con il comando seguente:

```
$ pdflatex "&pdfxmltex" esempio-014.fo [ Invio ]
```

Nel caso dovessero essere presenti riferimenti incrociati, sarebbe necessario eseguire la composizione con LaTeX per tre volte di seguito.

Ecco il risultato della composizione:



52.8 Riferimenti

- W3C, *Extensible Markup Language (XML) 1.0*, http://www.w3.org/TR/REC-xml/
- James Clark, *Comparison of SGML and XML*, http://www.w3.org/TR/NOTE-sgml-xml-971215.html
- Peter Flynn, *The XML FAQ*, http://xml.silmaril.ie/
- Norman Walsh, *A Technical Introduction to XML*, http://nwalsh.com/docs/articles/xml/
- W3C, *Namespaces in XML*, http://www.w3.org/TR/REC-xml-names/
- James Clark, *XML namespaces*, http://www.jclark.com/xml/xmlns.htm
- O'Reilly, *xml.com*, http://www.xml.com
- *Cover Pages*, http://xml.coverpages.org/xml.html
- W3C, *XSL Transformations (XSLT)*, http://www.w3.org/TR/xslt
- W3C, *XML Path Language (XPath)*, http://www.w3.org/TR/xpath
- Elliotte Rusty Harold, *XML Bible*, http://rf4.us/lib/old/books/programming/XMLBible.pdf
- Michele Sciabarrà, *Tutorial XML/XSLT*, http://www.corsojava.it/testi/xml/
- Michele Sciabarrà, *Linux e programmazione web*, McGraw-Hill, 2001, ISBN 88-386-4177-3; in particolare il capitolo *XML e XSLT*
- Apache software foundation, *FOP*, http://xmlgraphics.apache.org/fop/
- W3C, *Extensible stylesheet language (XSL)*, http://www.w3.org/TR/xsl/
- W3C, *Extensible stylesheet language*, http://www.w3.org/TR/xsl/
- Herong Yang, *Herong's notes on XSL-FO and XHTML*, http://www.herongyang.com/xhtml/
- David Carlisle, *xmltex: A non validating (and not 100% conforming) namespace aware XML parser implemented in TeX**, http://www.dcarlisle.demon.co.uk/xmltex/manual.html

- Sebastian Rahtz, *PassiveTeX*, http://wayback.archive.org/web/*/http://www.tei-c.org/Software/passivetex

[1] **Xalan** software libero con licenza speciale

[2] **FOP** software libero con licenza speciale

[3] Kaffe è un vecchio interprete di applicazioni Java, sviluppato prima che fosse rilasciato il pacchetto OpenJDK con una licenza libera. Attualmente, le distribuzioni GNU/Linux comuni dovrebbero disporre di una «macchina virtuale» Java, ed eventualmente di un ambiente di sviluppo Java, attraverso OpenJDK, GCJ e altri componenti liberi, senza complicazioni per l'utilizzatore.

[4] **XMLTeX** LPPL

[5] **PassiveTeX** Software libero con licenza speciale

LinuxDoc, DocBook e TEI

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] [-O] [--] file_sgml
```

```
debiandoc2lout [-k] [-O] [--] file_sgml
```

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

```
debiandoc2texinfo [-k] [-O] [--] file_sgml
```

```
debiandoc2text [-k] [-O] [--] file_sgml
```

```
debiandoc2textov [-k] [-O] [--] 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.

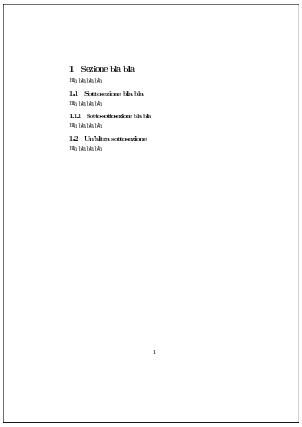
Una divisione ha probabilmente un titolo, che si definisce utilizzando l'elemento '**head**' prima del testo che si vuole inserire al suo interno:

```
<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**':

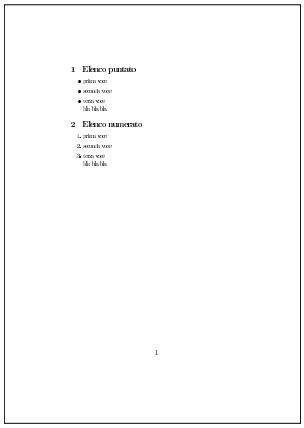
```
<list type="simple"|"bulleted"|"ordered">
    <item>testo_lineare|blocco</item>
    …
</list>
```

```
<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.

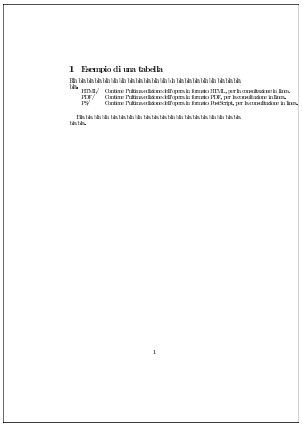
Le celle della tabella possono contenere sia testo lineare, sia blocchi; tuttavia, può darsi che il sistema di composizione non sia in grado di gestire celle che contengono testo molto lungo.

```
<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>
```



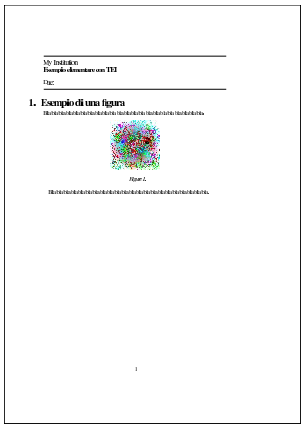
```
        <fileDesc>
            <titleStmt>
                <title>Esempio elementare con TEI</title>
            </titleStmt>
            <publicationStmt>
                <p/>
            </publicationStmt>
            <sourceDesc>
                <p/>
            </sourceDesc>
        </fileDesc>
    </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**'.

HTML

54.1 URI e IRI

Un URI (*Uniform resource identifier*) è un indirizzo espresso attraverso una stringa di caratteri per identificare una risorsa fisica o astratta. La risorsa in questione è un'entità e la sua collocazione non si trova necessariamente all'interno di una rete. In pratica, il concetto di URI incorpora i concetti di URL (*Uniform resource locator*) e di URN (*Uniform resource name*).

Un URL identifica una risorsa rappresentando il metodo di accesso a questa; un URN identifica la risorsa attraverso un nome, il quale deve essere unico a livello globale e deve persistere anche quando la risorsa cessa di esistere o diventa inaccessibile.

Un IRI (*Internationalized resource identifier*) è un URL che consente però l'uso dei caratteri previsti da Unicode, ovvero dalla codifica universale.

54.1.1 Trascrivibilità

L'esigenza primaria degli indirizzi URI e degli IRI è la loro «trascrivibilità». Con questo termine si vuole fare riferimento alla facilità con la quale questi devono poter essere trascritti, sia a livello meccanico, sia a livello umano. In pratica:

- un URI o un IRI è composto da una sequenza di «caratteri» e non necessariamente da ottetti (byte);
- un URI o un IRI deve poter essere trascritto attraverso qualunque mezzo, come una pubblicazione stampata o un appunto fatto a mano, in tal senso non può utilizzare caratteri particolari che possono mancare in un contesto determinato;
- un URI o un IRI deve poter essere ricordato facilmente dalle persone, per cui è utile che la stringa che rappresenta un URI abbia un significato che ne faciliti la memorizzazione.

Dal momento che ci deve essere la possibilità di rappresentare un URI o un IRI all'interno di parentesi di qualsiasi tipo, i caratteri corrispondenti a queste parentesi non possono essere utilizzati letteralmente all'interno di un indirizzo del genere. Le parentesi in questione sono quelle tonde, quadre, graffe e angolari: '**(**', '**)**', '**[**', '**]**', '**{**', '**}**', '**<**', '**>**'.

54.1.2 Sintassi

La sintassi di un URI o un IRI è piuttosto complessa, perché dipende molto dal contesto a cui si applica. Non è il caso si entrare troppo nel dettaglio; piuttosto è meglio apprendere la logica della cosa.

schema **:** ***parte_successiva_dipendente_dallo_schema***

Quello che si vede è il modello di prima approssimazione di un indirizzo URI o IRI assoluto (viene trattato in seguito il concetto di URI/IRI relativo). In questa prima fase si distinguono due parti, separate da due punti verticali ('**:**'), dove prima appare un nome che definisce uno «schema» e poi continua con una stringa che va interpretata in base alle regole specifiche di quello schema.

La sintassi di un URI/IRI non stabilisce a priori quale sia la forma che deve avere la stringa che segue i due punti; tuttavia, è frequente l'utilizzo di URI secondo i modelli seguenti:

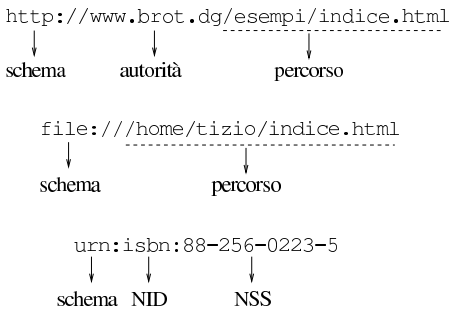
schema **:** ***//autorità*** **[** ***percorso*** **[** **?** ***interrogazione*** **]]**

schema **:** ***/percorso***

Convenzionalmente, quando una risorsa viene individuata attraverso un URI/IRI che per sua natura contiene un'informazione gerarchica, la separazione tra i vari livelli di questa gerarchia avviene utilizzando una barra obliqua normale ('**/**'). Si tratta evidentemente di una tecnica ereditata dal file system Unix; tuttavia, ciò resta indipendente dal fatto che la risorsa in questione risieda fisicamente all'interno di un file system o meno.

La figura 54.1 mostra alcuni esempi a proposito di URI/IRI composti secondo i modelli più frequenti.

Figura 54.1. Esempi di URI/IRI comuni.



Nella figura si vede anche un caso particolare, riferito a un URN di tipo ISBN (*International standard book number*). Lo schema di un URN è sempre '**urn:**'; a questo segue l'indicazione di un NID (*Namespace identifier*), ovvero un identificatore che qualifica l'informazione successiva; infine si inserisce l'informazione, definita NSS (*Namespace specific string*), ovvero ciò che va inteso nel contesto stabilito dal NID. L'esempio che appare nella figura fa riferimento al numero ISBN 88-256-0223-5, esprimendolo in forma di URN.

54.1.2.1 Accesso a un servente attraverso la rete

Quando l'indirizzo URI si riferisce a un servizio offerto attraverso la rete, la struttura di ciò che è stato definito come «autorità» si articola in modo particolare:

[***utente*** **[** **:** ***parola_d'ordine*** **]@]** ***nodo*** **[** **:** ***porta*** **]**

In questo modo si può specificare il nominativo utente per l'accesso alla risorsa, eventualmente anche la parola d'ordine (benché ciò sia decisamente sconsigliabile per motivi di sicurezza), quindi il nodo di rete che offre il servizio e infine la porta del servizio.

Il nodo di rete può essere indicato per nome, attraverso il nome a dominio, oppure attraverso il numero IPv4. Purtroppo non è sta-

to definito un modo per indicare un numero IPv6, dal momento che la sua forma renderebbe impossibile l'interpretazione corretta dell'indirizzo.

Se si omettono le informazioni riferite all'utente, vuol dire che queste non sono necessarie, oppure che esistono dei valori predefiniti per questo; per quanto riguarda la porta del servizio, se questa non viene indicata si fa riferimento sempre al suo valore predefinito. Naturalmente, è stabilito dal servente quali siano i valori predefiniti.

54.1.2.2 Riferimento agli URI

Per sua natura, l'indirizzo URI è un riferimento a una risorsa. In generale vanno considerate anche due circostanze particolari: il riferimento a un frammento della risorsa e l'indicazione di URI relativi.

Un URI relativo è un indirizzo ridotto che parte da un punto di partenza conosciuto. Il principio deriva dal concetto di percorso relativo all'interno di un file system. In generale, un URI relativo può essere indicato omettendo tutta la parte iniziale che si possa determinare altrimenti.

Di fronte a un URI che contenga un'informazione sul percorso in forma gerarchica, è abbastanza facile intendere cosa sia la base di riferimento per gli URI relativi: basta togliere dall'indirizzo attuale tutto quello che segue l'ultima barra obliqua. Per esempio, per il documento *http://www.brot.dg/esempi/articolo.html* l'URI di base è *http://www.brot.dg/esempi/*, per cui, il riferimento a '`figure/foto.jpg`' richiama effettivamente l'URI *http://www.brot.dg/esempi/figure/foto.jpg*.

Il percorso di un URI relativo può essere indicato anche con una barra obliqua iniziale, ma in questo caso si intende fare riferimento a un percorso assoluto nell'ambito dell'URI. Continuando con l'esempio precedente, il riferimento a '`/nuovo/documento.html`' richiama effettivamente l'URI *http://www.brot.dg/nuovo/documento.html*.

In presenza di un percorso relativo, è possibile utilizzare anche i simboli '`.`' e '`..`', con lo stesso significato che hanno nel file system Unix: il primo rappresenta la posizione corrente e il secondo quella precedente.

È importante osservare che il riferimento alla stringa nulla indica implicitamente lo stesso URI iniziale.

Il problema degli URI relativi non è così semplice come è stato descritto. In realtà vanno prese in considerazione altre cose, come per esempio la possibilità che il tipo di risorsa (di solito in un documento HTML) possa incorporare l'informazione esplicita di un URI di base.

Quando il tipo di risorsa lo consente, è possibile aggiungere all'URI l'indicazione di un frammento particolare. Questa parte aggiuntiva la si riconosce perché è preceduta dal simbolo '`#`':

```
http://www.brot.dg/esempi/articolo.html#commento
```

L'esempio mostra il riferimento al frammento '**`#commento`**' nell'ambito dell'URI '`http://www.brot.dg/esempi/articolo.html`'. Dal momento che la stringa nulla fa riferimento alla risorsa attuale, i riferimenti interni alla stessa risorsa sono indicati facilmente attraverso il solo frammento:

```
#commento
```

L'esempio mostra un riferimento relativo al frammento '**`#commento`**' della risorsa corrente.

54.1.2.3 Esempi

Frequentemente, il nome dello schema dell'indirizzo URI corrisponde al nome del protocollo necessario per raggiungere la risorsa relativa. I più comuni sono: '**`http`**', '**`ftp`**', '**`gopher`**', '**`mailto`**', '**`wais`**', '**`telnet`**', '**`tn3270`**', '**`news`**'. Quando si vuole fare riferimento a un file locale senza utilizzare alcun protocollo particolare, si può indicare anche lo schema '**`file`**', ma in questo caso ci sono delle particolarità che vengono mostrate dagli esempi.

- `http://www.brot.dg:8080/esempi/indice.html`
  - protocollo HTTP
  - nodo di rete *www.brot.dg*
  - porta 8080
    Viene indicata la porta perché si vuole fare riferimento a un valore diverso dallo standard che per il protocollo HTTP è 80
  - risorsa '`/esempi/indice.html`'
- `http://www.brot.dg/esempi/indice.html`
  Come nell'esempio precedente, ma senza l'indicazione della porta che questa volta corrisponde al valore predefinito, cioè 80.
- `http://192.168.1.1/esempi/indice.html`
  Come nell'esempio precedente, ma l'indicazione del nodo avviene per mezzo del suo indirizzo IPv4 invece che attraverso il nome a dominio.
- `ftp://ftp.brot.dg/pub/archivi/esempio.tar.gz`
  - protocollo FTP
  - nodo di rete *ftp.brot.dg*
  - risorsa '`/pub/archivi/esempio.tar.gz`'
- `ftp://tizio@ftp.brot.dg/pub/archivi/esempio.tar.gz`
  Come nell'esempio precedente, con la differenza che si fa riferimento a un utente particolare.
- `ftp://tizio:segretissima@ftp.brot.dg/pub/archivi/esempio.tar.gz`
  Come nell'esempio precedente, con la differenza che si aggiunge l'indicazione della parola d'ordine di accesso al servizio, cosa che in generale è bene non passare mai in questo modo.
- `file://localhost/home/daniele/indice.html`
  In questo caso si vuole fare riferimento a un file locale. Precisamente si tratta del file '`/home/daniele/indice.html`' contenuto nell'elaboratore *localhost*.
  Questo tipo di indicazione è utile specialmente quando si vuole fare riferimento a una pagina indice o iniziale, caricata automaticamente all'atto dell'avvio di un programma cliente per la navigazione.
- `file:///home/daniele/indice.html`
  Esattamente come nell'esempio precedente, con la differenza che si omette l'indicazione esplicita dell'elaboratore locale: *localhost*.
- `file:/home/daniele/indice.html`
  Esattamente come nell'esempio precedente, con la differenza che si utilizza una sola barra obliqua dopo l'indicazione '**`file:`**' (ma in generale è preferibile la forma precedente, con le tre barre oblique).
- `mailto:tizio@dinkel.brot.dg`
  Si tratta di un indirizzo di posta elettronica, nel quale è essenziale fornire l'indicazione del nominativo utente. Dopo il nome del nodo di destinazione non appare un percorso, perché in questo caso non avrebbe significato.

54.1.3 Limitazioni nell'uso dei caratteri

Ogni componente di un URI ha delle regole proprie nell'uso dei caratteri, dal momento che alcuni di questi hanno significati speciali. Purtroppo le regole in questione sono tante e la cosa migliore che si può fare è quella di usare il buon senso, riservando la lettura della documentazione specifica ai casi in cui è indispensabile chiarire il problema nel dettaglio (RFC 2396).

In generale non è ammissibile l'uso dello spazio. Infatti, considerato il principio di trascrivibilità degli URI, lo spazio dovrebbe essere inteso solo come una necessità legata al tipo di trascrizione utilizzata. Per il resto, se la propria lingua lo consente, sarebbe bene limitarsi all'uso delle lettere dell'alfabeto latino (maiuscole e minuscole, ma senza accenti), le cifre numeriche e alcuni simboli: '**@**', '*****', '**_**', '**-**' e il punto ('**.**'). Gli altri simboli possono creare problemi di trascrivibilità o avere significati particolari (basta pensare alle barre oblique e ai due punti verticali).

Quando un simbolo particolare non può essere utilizzato in modo letterale nel contesto in cui lo si vuole inserire, può essere indicato attraverso una notazione speciale: '***%hh***'. La sigla ***hh*** rappresenta una coppia di cifre esadecimali. A questa regola fa eccezione lo spazio che viene codificato normalmente con il segno '**+**', ma non in tutte le occasioni (di solito solo nelle stringhe di richiesta).

Generalmente, per gli indirizzi URI normali non c'è la necessità di preoccuparsi di questo problema, anche la tilde può essere utilizzata letteralmente nell'indicazione dei percorsi. La tabella 54.4 mostra l'elenco di alcune corrispondenze tra simboli particolari e la codifica alternativa utilizzabile negli URI.

Tabella 54.4. Alcune corrispondenze tra simboli particolari e codifica alternativa utilizzabile negli URI.

Carattere	Codifica corrispondente
%	%25
&	%26
+	%2B
/	%2F
=	%3D

In linea di principio, un URI dovrebbe essere realizzato in modo da non dover utilizzare questa tecnica di protezione per i caratteri «speciali». La situazione più probabile in cui è necessario utilizzare questo procedimento è riferito alle stringhe di interrogazione.

54.2 Aspetti generali di HTML

HTML sta per *Hypertext markup language* e in pratica è un formato SGML per i documenti della rete che fa uso di un DTD particolare: HTML appunto. La composizione di un documento HTML non può mai essere valutata perfettamente in anticipo, perché dipende da diversi fattori:

- il programma utilizzato per visualizzare il documento;
- la risoluzione utilizzata;
- i tipi di carattere a disposizione;
- la profondità di colori disponibili.

Lo standard HTML è tale per cui tutti (o quasi) i programmi utilizzabili per la lettura di tali documenti sono in grado di cavarsela. Ma questo risultato minimo è ben lontano dall'esigenza di costruire qualcosa che tutti possano vedere più o meno nello stesso modo. Per questo, quando si costruisce un documento HTML, occorre pensare all'utenza a cui è destinato, in modo da decidere quali caratteristiche possono essere utilizzate e quali invece è meglio scartare per evitare inutili problemi di lettura.

L'HTML nasce all'inizio degli anni 1990, abbinato in particolare al primo navigatore: Mosaic. Da quel momento a oggi il formato

HTML ha subito diversi aggiornamenti; si ricorda in particolare la versione 2.0 del 1995 e la versione 3.2 del 1997. Secondo il progetto di W3C, lo sviluppo di HTML avrebbe dovuto concludersi con la versione 4.01, per passare da quel momento a XHTML. Tuttavia, intorno al 2007 la questione di HTML viene riaperta, introducendo una versione 5. Tra le varie versioni di HTML va tenuta in considerazione particolare quella definita dallo standard ISO 15445, la quale può considerarsi un passepartout per ogni tipo di navigatore ipertestuale.

Lo spirito alla base dello sviluppo dell'HTML da parte del W3C, così come acquisito dallo standard ISO 15445, è quello di ottenere un formato multimediale-ipertestuale completo, adatto per la lettura attraverso qualunque tipo di mezzo: dal terminale tattile braille al documento stampato. Le estensioni proprietarie di questo standard si sono rivolte principalmente all'aspetto visuale e scenografico del formato, trascurando le altre esigenze. Scrivere un documento «puro» in HTML è un'arte raffinata, ma poco conosciuta. In generale, **maggiori sono i contenuti e le esigenze di divulgazione, minori devono essere le pretese estetiche**.

La documentazione di riferimento per tutto ciò che riguarda l'HTML è quella offerta dal W3C: *http://www.w3.org* , in particolare *http://www.w3.org/TR/* , a cui si deve affiancare anche quanto riguarda lo standard ISO 15445: *http://www.scss.tcd.ie/misc/15445/15445.html* .

54.2.1 HTML e SGML

L'HTML è un linguaggio di composizione basato sull'SGML (si veda quanto descritto nel capitolo 51). Come tale, un documento HTML inizia sempre con la dichiarazione del DTD; poi tutto il documento viene racchiuso nell'elemento principale di questa struttura:

```
<!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
<HTML>
    ...
    ...
    ...
</HTML>
```

Purtroppo, la maggior parte dei programmi di navigazione o di composizione per il formato HTML non è in grado di comprendere tutte le regole dell'SGML, per cui occorre evitare di utilizzare alcune delle sue caratteristiche. In particolare bisogna evitare:

- la creazione di entità generali per l'utilizzo di macro specifiche relative al testo;

```
<!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN"
[
<!ENTITY GNULINUX "GNU/Linux">
<!ENTITY HURD     "GNU/Hurd">
<!ENTITY DOS      "Dos">
<!ENTITY POSIX    "POSIX">
<!ENTITY UNIX     "Unix">
]>
```

- le sezioni marcate per l'inclusione e l'esclusione del testo;

```
<![INCLUDE[
    …
    <!-- testo incluso -->
    …
]]>
…
<![IGNORE[
    …
    <!-- testo escluso -->
    …
]]>
```

- le sezioni marcate per individuare un contenuto di tipo '**CDATA**', allo scopo di proteggere il simbolo '**<**' in modo da poterlo usare letteralmente;

```
<![CDATA[
    …
    <!-- testo letterale -->
    …
]]>
```

• la delimitazione di un elemento in forma abbreviata;

```
<nome_elemento / contenuto_dell'elemento /
```

• l'indicazione di marcatori iniziali e finali vuoti.

```
<> ... </>
```

Il fatto che l'HTML sia definito da un DTD, permette di verificare la sua correttezza formale, anche se le regole stabilite nel DTD non sono sufficienti a definire la sintassi completa. Per poter verificare la correttezza formale di un documento HTML, oltre agli strumenti di convalida, cioè il pacchetto SP, occorre procurarsi il DTD e le sue estensioni riferite alle entità generali, quelle che permettono di utilizzare le macro per le lettere accentate e i simboli speciali.

Il DTD dell'HTML ISO 15445 e la dichiarazione SGML si trovano presso *https://www.scss.tcd.ie/misc/15445/15445.HTML* (da cui vanno estrapolati i file '`15445.dcl`' e '`15445.dtd`'). Per quanto riguarda le entità standard a cui si fa riferimento, queste si trovano presso *http://www.w3.org/TR/html4/* . Si può realizzare un catalogo SGML per l'analisi locale di un documento del genere nel modo seguente:

```
OVERRIDE YES

SGMLDECL                                                    15445.dcl

PUBLIC  "ISO/IEC 15445:2000//DTD HyperText Markup Language//EN" 15445.dtd
DTDDECL "ISO/IEC 15445:2000//DTD HyperText Markup Language//EN" 15445.dcl
PUBLIC  "ISO/IEC 15445:2000//DTD HTML//EN"                  15445.dtd
DTDDECL "ISO/IEC 15445:2000//DTD HTML//EN"                  15445.dcl
PUBLIC "-//W3C//ENTITIES Full Latin 1//EN//HTML"            HTMLlat1.ent
PUBLIC "-//W3C//ENTITIES Special//EN//HTML"                 HTMLspecial.ent
PUBLIC "-//W3C//ENTITIES Symbolic//EN//HTML"                HTMLsymbol.ent
```

Con questo catalogo, i file utilizzati devono trovarsi nella directory corrente.

Nell'esempio seguente si utilizza il programma '**`nsgmls`**' (del pacchetto SP) supponendo in particolare che il catalogo sia contenuto nel file '`catalogo`'; il file da verificare viene indicato come '`mio_file.html`'. Il catalogo e il file da controllare si intendono collocati nella directory corrente.

```
$ cat mio_file.html | nsgmls -s -c catalogo [Invio]
```

È il caso di ricordare che alcune distribuzioni GNU/Linux, in particolare Debian, predispongono un pacchetto apposito contenente i DTD più comuni riferiti alle varie versioni dell'HTML, comprese le estensioni proprietarie, assieme alle relative entità standard. Naturalmente, il tutto è organizzato in un catalogo unico che va eventualmente ad aggiornare il catalogo di sistema (dovrebbe trattarsi del file '`/etc/sgml.catalog`', oppure del file '`/usr/share/sgml/catalog`'). Il nome di questo pacchetto potrebbe essere '`sgml-data*`'.

Oltre alla verifica in base al DTD sarebbe opportuno sapere leggere il contenuto del DTD stesso. A questo proposito è da notare il fatto che nel manuale che descrive le specifiche HTML ISO 15445, si fa spesso riferimento alle caratteristiche degli elementi attraverso lo schema offerto dalla dichiarazione relativa nel DTD. In effetti, ciò permette di rendere molto chiara e precisa la descrizione che ne viene fatta subito dopo.

54.2.1.1 Attributi comuni attraverso le entità parametriche

Il DTD dell'HTML ISO 15445 fa un uso massiccio di entità parametriche e questo può disorientare inizialmente. In generale basta ricordare che qualunque cosa nella forma '***%nome;***' è una macro che si espande in una stringa. La dichiarazione di queste entità parametriche avviene nella parte iniziale del DTD, attraverso istruzioni del tipo:

```
<!ENTITY % nome "stringa">
```

È interessante notare l'utilizzo di entità parametriche per fare riferimento agli attributi degli elementi. Infatti, quasi tutti gli elementi

dell'HTML ISO 15445 prevedono l'uso di attributi, per cui si è ritenuto opportuno classificarli all'interno di entità parametriche. In particolare è importante individuarne due molto importanti:

```
<!ENTITY % core
   "CLASS      CDATA      #IMPLIED -- Comma separated list of class values --
    --The name space of the ID attribute is shared with the name space of
      the NAME attributes.  Both ID and NAME attributes may be provided for
      the <A> and <MAP> elements. When both ID and NAME values are provided
      for an element, the values shall be identical.  It is an error for an
      ID or NAME value to be duplicated within a document.

      It is recommended that authors of documents specify both the ID
      attribute and the NAME attribute for the <A> and <MAP> elements.
    --
    ID         ID         #IMPLIED -- Document-wide unique id --
    TITLE      CDATA      #IMPLIED -- Advisory title or amplification --" >

        <!-- Internationalization attributes -->

<!ENTITY % i18n
   "DIR        (ltr|rtl)  #IMPLIED -- Direction for weak/neutral text --
    LANG       NAME       #IMPLIED -- RFC1766 language value --" >
```

La macro '**`%core;`**' serve a individuare un gruppo di attributi disponibili nella maggior parte degli elementi:

Attributo	Descrizione
`ID`	permette di attribuire una stringa di riconoscimento all'elemento, in modo da potervi fare riferimento;
`CLASS`	permette di abbinare all'elemento una classe, definita attraverso un nome, in modo da potergli attribuire uno stile particolare;
`TITLE`	permette di attribuire un «titolo» all'elemento, cosa che si traduce in pratica in modo differente a seconda del contesto (ovvero, a seconda dell'elemento a cui si applica).

La macro '**`%i18n;`**' serve invece a definire ciò che riguarda la localizzazione:

Attributo	Descrizione
`LANG`	permette di indicare una sigla, secondo lo standard ISO 639 (tabella 13.4) e anche secondo altri standard, per attribuire all'elemento il linguaggio relativo;
`DIR`	permette di stabilire il flusso del testo nel risultato finale, dove la parola chiave '**`ltr`**' si riferisce a uno scorrimento da sinistra a destra (*Left to right*) e la parola chiave '**`rtl`**' indica uno scorrimento opposto, da destra a sinistra (*Right to left*).

Si osservi, a titolo di esempio, la dichiarazione dell'elemento '**`P`**', dove gli attributi sono quelli più comuni, rappresentati dalle macro '**`%core;`**' e '**`%i18n;`**':

```
<!ELEMENT P     - O  (%text;)+ >
<!ATTLIST P
    %core;          -- Element CLASS, ID and TITLE --
    %i18n;          -- Internationalization DIR and LANG -->
```

54.2.1.2 Classificazione fondamentale degli elementi

All'interno di un documento HTML si distinguono due gruppi di elementi fondamentali: quelli che rappresentano dei blocchi e quelli che servono a inserire qualcosa all'interno di una riga di testo normale. Questa suddivisione corrisponde a due macro: '**`%block;`**' e '**`%text;`**' rispettivamente.

Per fare un esempio, l'elemento '**`P`**' (paragrafo) è un «blocco», mentre l'elemento '**`EM`**' (enfasi) è un componente interno a una riga di testo.

Questa classificazione semplifica molto la dichiarazione degli elementi, come nel caso dell'elemento '**`P`**', già visto, il cui contenuto è semplicemente tutto ciò che va inserito nelle righe di testo:

```
<!ELEMENT P     - O  (%text;)+ >
```

Alcuni elementi di un documento HTML sono ambigui, nel senso che possono contenere sia blocchi che testo. A titolo di esempio si

osservi la dichiarazione dell'elemento '**LI**' che rappresenta la voce di un elenco puntato o numerato:

```
<!ELEMENT LI      - O  (%text; | %block;)+ >
```

54.2.2 Stili

Le estensioni proprietarie dell'HTML hanno portato questo linguaggio di composizione a una proliferazione di dialetti, a causa dell'esigenza di trasferire anche le informazioni sull'aspetto finale della composizione, che in origine non sono state prese in considerazione. L'unica soluzione disponibile con HTML ISO 15445 è l'abbinamento di uno stile, che può essere dichiarato all'interno del file HTML stesso, attraverso l'elemento '**STYLE**', oppure in un file esterno, richiamandolo con l'elemento '**LINK**' (viene mostrato tra poco).

L'HTML non presuppone il formato in cui può essere realizzato lo stile. È comune l'uso di stili in formato CSS (*Cascading style sheet*) e per farvi riferimento si indica il tipo '**text/css**'.

Per il momento, non viene spiegato in che modo si scrivono le direttive in un foglio di stile CSS. Intuitivamente, il lettore può comprendere che la direttiva seguente serve a colorare in blu il contenuto degli elementi '**H1**':

```
H1 { color: blue; }
```

Inoltre, la direttiva seguente serve per fare in modo che il contenuto dell'elemento '**P**' abbia il carattere di 12 punti e di colore rosso:

```
P { font-size: 12pt; color: red; }
```

Si osservi che la stessa cosa avrebbe potuto essere scritta nel modo seguente:

```
P {
    font-size:  12pt;
    color:      red;
}
```

Per definire questi stili all'interno di un documento HTML, senza fare uso di un file esterno, si potrebbe agire nel modo seguente, attraverso l'uso dell'elemento '**STYLE**':

```
<!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
<HTML>
<HEAD>
    <TITLE>Esempio</TITLE>
    <STYLE TYPE="text/css">
        H1 { color: blue }
        P {
            font-size:  12pt;
            color:      red;
        }
    </STYLE>
</HEAD>
<BODY>
    ...
    ...
    ...
</BODY>
</HTML>
```

Si comprende che il testo contenuto nell'elemento '**STYLE**' non deve interferire con l'HTML e quindi non può contenere simboli che possano risultare ambigui. Questo problema riguarda naturalmente il linguaggio con cui è realizzato lo stile; nel caso del formato CSS non dovrebbe porsi alcun problema. Tuttavia, qualche programma utilizzato per la navigazione, potrebbe non riconoscere l'elemento '**STYLE**', arrivando a riprodurre il testo che rappresenta in realtà lo stile. Per evitare questo problema si può circoscrivere la cosa all'interno di un commento SGML:


```
<STYLE TYPE="text/css">
    <!--
        H1 { color: blue }
        P {
            font-size:  12pt;
            color:      red;
        }
    -->
</STYLE>
```

Probabilmente, il modo più elegante di abbinare uno stile a un documento HTML è quello di aggiungere un file esterno. Nell'esempio seguente si include lo stile corrispondente al file '`stile.css`':

```
<!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
<HTML>
<HEAD>
    <TITLE>Esempio</TITLE>
    <LINK REL="stylesheet" TYPE="text/css" HREF="stile.css">
    ...
</HEAD>
...
</HTML>
```

È chiaro che dipende dal programma di navigazione la capacità o meno di conformarsi allo stile. In generale, lo standard CSS sembra essere quello che ha più probabilità di affermarsi.

54.2.3 Struttura di un documento HTML

Il documento HTML è contenuto tutto nell'elemento omonimo: '**HTML**'. Questo si scompone in due elementi fondamentali, '**HEAD**' e '**BODY**', che rappresentano rispettivamente l'intestazione e il corpo:

```
<!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
<HTML>
<HEAD>
    <TITLE>Titolo della pagina</TITLE>
</HEAD>
<BODY>
    ...
    <!-- Corpo del documento -->
    ...
</BODY>
</HTML>
```

In generale, è conveniente annotare la lingua principale del documento, attraverso l'attributo '**LANG**' da collocare nel marcatore di apertura dell'elemento '**HTML**':

```
<HTML LANG="it">
```

Per la precisione, il codice che definisce il linguaggio viene indicato secondo la sintassi seguente:

codice_principale [- ***codice_secondario***]

In pratica, la prima parte, quella che appare prima del trattino di separazione, indica la lingua, di solito attraverso il codice ISO 639 (tabella 13.4), mentre la seconda parte indica l'area nazionale, secondo lo standard ISO 3166 (tabella 13.5), che a sua volta può implicare delle varianti nel linguaggio.

In generale, un documento di grandi dimensioni realizzato attraverso il formato HTML, richiede la scomposizione dello stesso in più file HTML collegati tra loro da riferimenti ipertestuali. Questa, purtroppo, è una necessità a causa delle limitazioni dei programmi di navigazione.

54.2.3.1 Intestazione e informazioni supplementari

L'intestazione è una parte del documento HTML che serve per annotare delle informazioni generali. Deve contenere almeno il titolo all'interno dell'elemento '**TITLE**'. Di solito, la riproduzione di un documento HTML non fa apparire il titolo nel testo del documento, che comunque viene usato per farvi riferimento (per esempio nel segnalibro del programma utilizzato per la sua visualizzazione).

Nell'intestazione, prima o dopo il titolo, può essere conveniente collocare alcune «meta-informazioni», attraverso alcuni elementi '**META**'. Si tratta di un elemento vuoto, per il quale si utilizza soltanto il marcatore di apertura con l'indicazione di attributi opportuni. In particolare, si possono utilizzare gli attributi seguenti:

Attributo	Descrizione
NAME	per indicare un nome che qualifica il tipo di meta-informazione (si tratta di parole chiave più o meno standard, che però non sono state definite nel DTD);
HTTP-EQUIV	per indicare un campo di risposta nell'ambito del protocollo HTTP, tenendo conto che l'attributo '**NAME**' è alternativo a '**HTTP-EQUIV**';
CONTENT	(obbligatorio) per indicare il valore abbinato al nome indicato attraverso l'attributo '**NAME**', oppure attraverso l'attributo '**HTTP-EQUIV**'.

Come si intuisce dall'elenco degli attributi più importanti, si può distinguere tra elementi '**META**' che utilizzano l'attributo '**NAME**' e altri che usano l'attributo '**HTTP-EQUIV**'. Le informazioni che si definiscono attraverso elementi '**META**' con l'attributo '**NAME**' permettono di indicare informazioni che qualificano il documento, soprattutto quando questo viene trattato automaticamente da un motore di ricerca; l'attributo '**HTTP-EQUIV**' permette invece di intervenire a livello del protocollo HTTP (quando il documento viene ottenuto in questo modo), specificando le intestazioni HTTP relative. Si osservi l'esempio seguente:

```
<HEAD>
    <TITLE>Titolo della pagina</TITLE>
    <META HTTP-EQUIV="Content-Type"
          CONTENT="text/html; charset=UTF-8">
    <META NAME="Description"
          CONTENT="Esempio di una pagina HTML">
    <META NAME="Keywords"
          CONTENT="HTML, SGML, Editoria elettronica">
    <META NAME="Author"
          CONTENT="P. Pallino ppallino@dinkel.brot.dg">
    <META NAME="Classification" CONTENT="Esempio HTML">
</HEAD>
```

In particolare, ricevendo questo documento attraverso il protocollo HTTP, si ottiene anche l'intestazione HTTP seguente:

```
Content-Type: text/html; charset=UTF-8
```

Si noti nell'esempio l'indicazione esplicita dell'insieme di caratteri: UTF-8.

Un altro tipo di elemento speciale può apparire all'interno dell'intestazione di un documento HTML; si tratta di '**LINK**'. Anche questo è un elemento vuoto e serve solo per indicare degli attributi nel marcatore di apertura. Gli attributi più importanti sono:

Attributo	Descrizione
HREF	per indicare un URI a cui si intende fare riferimento;
REL	per definire la relazione che c'è con questo tipo di collegamento;
TYPE	per specificare in anticipo il tipo dei dati contenuti nell'URI;
MEDIA	per specificare il mezzo attraverso cui viene letto il documento.

Trattandosi di un elemento vuoto, collocato nell'intestazione HTML, non è pensato per essere rappresentato nella composizione. Tuttavia, abbinando le parole chiave opportune all'attributo '**REL**', si stabiliscono dei collegamenti utili per ricomporre un documento più grande costituito da più pagine HTML. In pratica, si può dichiarare in modo esplicito come è articolato, così che il programma di navigazione o composizione sappia regolarsi. La tabella 54.29 elenca alcune delle parole chiave che possono essere assegnate all'attributo '**REL**'.

Tabella 54.29. Parole chiave tipiche da assegnare all'attributo '**REL**' dell'elemento '**LINK**'.

Nome	Descrizione
Alternate	Una versione alternativa dello stesso documento.
Stylesheet	Foglio di stile esterno.
Start	Il primo documento di una collezione.
Next	Il prossimo documento di una sequenza lineare.
Prev	Il documento precedente di una sequenza lineare.
Contents	Un documento che funge da indice generale.
Index	Un documento che funge da indice analitico.
Glossary	Un documento che funge da glossario.
Copyright	Un documento che contiene la dichiarazione del copyright.
Chapter	Un documento che funge da capitolo in una collezione.
Section	Un documento che funge da sezione in una collezione.
Subsection	Un documento che funge da sottosezione in una collezione.
Appendix	Un documento che funge da appendice in una collezione.
Help	Un documento che funge da guida.

L'esempio seguente mostra parte di un'intestazione di una pagina HTML in cui sono stati usati alcuni elementi '**LINK**' per definire la relazione con altre pagine che compongono la stessa raccolta:

```
<HEAD>
    ...
    <LINK REL="Stylesheet" TYPE="text/css" HREF="stile.css">
    <LINK REL="Start" TITLE="inizio" HREF="index.html">
    <LINK REL="Contents" TITLE="indice generale"
          HREF="indice-generale.html">
    <LINK REL="Prev" TITLE="precedente"
          HREF="capitolo-6.html">
    <LINK REL="Next" TITLE="successivo"
          HREF="capitolo-8.html">
    <LINK REL="Index" TITLE="indice analitico"
          HREF="indice-analitico.html">
</HEAD>
```

Merita un po' di attenzione l'attributo '**MEDIA**' che serve a stabilire il mezzo adatto per la lettura del documento relativo. Questo attributo si usa generalmente all'interno di un elemento '**LINK**' che serve a indicare un foglio di stile esterno; inoltre può essere usato per lo stesso motivo all'interno di un elemento '**STYLE**'. In pratica, in questo modo, si stabilisce l'abbinamento tra stile e mezzo di lettura. La tabella 54.31 elenca i nomi che si possono assegnare a un attributo '**MEDIA**'.

Tabella 54.31. Parole chiave tipiche da assegnare all'attributo '**MEDIA**' dell'elemento '**LINK**' e dell'elemento '**STYLE**'.

Nome	Descrizione
screen	Schermo per lo scorrimento continuo.
tty	Terminale a celle di caratteri o simile.
tv	Televisione (bassa risoluzione e altre limitazioni).
projection	Proiettore.
handheld	Schermi portatili.
print	Stampa e simili (composizione impaginata).
braille	Terminale a barra braille per i non vedenti.
aural	Lettore a sintesi vocale.
all	Valido per tutti i tipi di dispositivo.

L'esempio seguente mostra in che modo si potrebbero selezionare diversi fogli di stile in base al mezzo utilizzato per la lettura del documento:

```
<LINK REL="Stylesheet" TYPE="text/css" MEDIA="screen"
      HREF="stile-schermo.css">
<LINK REL="Stylesheet" TYPE="text/css" MEDIA="tty"
      HREF="stile-testo.css">
<LINK REL="Stylesheet" TYPE="text/css" MEDIA="braille"
      HREF="stile-braille.css">
```

54.2.3.2 Corpo del documento

Il corpo di un documento HTML è delimitato dall'elemento '**BODY**' e il suo contenuto è ciò che alla fine viene mostrato nella composizione finale. La composizione del corpo viene descritta nella sezione 54.4.

54.3 Attributi comuni

Si è già accennato al fatto che molti elementi condividano un insieme comune di attributi. Vale la pena di descrivere brevemente alcuni di questi nelle sezioni successive.

54.3.1 Linguaggio

Il linguaggio di un elemento viene definito esplicitamente attraverso l'attributo '**LANG**', a cui viene assegnato solitamente un codice corrispondente allo standard ISO 639. La tabella 54.33 riporta un elenco di questi codici ridotto ad alcune lingue occidentali.

Tabella 54.33. Alcuni codici dello standard ISO 639 per la definizione della lingua attraverso una sigla di due soli caratteri.

Codice	Lingua	Codice	Lingua
fr	Francese	it	Italiano
ro	Rumeno	es	Spagnolo
ca	Catalano	co	Corso
pt	Portoghese	da	Danese
nl	Olandese	en	Inglese
de	Tedesco	is	Islandese
no	Norvegese	sv	Svedese
fi	Finlandese		

In generale può essere conveniente l'utilizzo di questo attributo nell'elemento '**HTML**', in modo da fissare il linguaggio di tutto il documento. Tuttavia, quando un elemento contiene un testo in un altro linguaggio, conviene annotarlo nello stesso modo.

L'effetto più evidente che potrebbe risultare dalla distinzione in base al linguaggio, è la separazione delle parole in sillabe, per creare una composizione più gradevole.

54.3.2 Codifica

L'opzione '**charset**' dell'attributo '**CONTENT**', permette di definire esplicitamente l'insieme di caratteri dell'elemento. Come è già stato mostrato, di solito lo si utilizza in un elemento '**META**' introduttivo allo scopo di definire l'intestazione HTTP relativa:

```
<HEAD>
    <TITLE>...</TITLE>
    <META HTTP-EQUIV="Content-Type"
          CONTENT="text/html; charset=UTF-8">
    <!--...-->
</HEAD>
```

La tabella 54.35 elenca alcuni codici comuni per la definizione dell'insieme dei caratteri.

Tabella 54.35. Alcuni codici per definire l'insieme di caratteri.

Codice	Corrispondenza	Codice	Corrispondenza
ISO−8859−1	latin1	ISO−8859−2	latin2
ISO−8859−3	latin3	ISO−8859−4	latin4
ISO−8859−5	ciryllic	ISO−8859−6	arabic
ISO−8859−7	greek	ISO−8859−8	hebrew
ISO−8859−9	latin5	ISO−8859−15	latin9
UTF-8	codifica universale compatibile con ASCII.	UTF-16	

54.3.3 Direzione del testo

Il testo di un documento HTML può scorrere da sinistra a destra o viceversa. Per controllare questo flusso si può utilizzare l'attributo '**DIR**', a cui si possono abbinare esclusivamente le parole chiave '**LTR**' o '**RTL**': *Left to right*, da sinistra a destra; *Right to left*, da destra a sinistra.

In generale, il flusso del testo avviene da sinistra a destra, come richiedono le lingue occidentali, per cui non è necessario usare questo attributo in condizioni «normali».

È importante notare che il testo nel sorgente di un documento HTML segue sempre il flusso normale, da sinistra a destra, ammesso che si possa definire un flusso per un file sorgente.

Non è disponibile la possibilità di ribaltare orizzontalmente i caratteri, quando il flusso del testo cambia direzione, come avviene nella scrittura geroglifica.

54.3.4 Titolo

Molti elementi dispongono di un attributo '**TITLE**'. Il suo scopo è quello di indicare un titolo, che viene preso in considerazione in modo differente in base al contesto. Questo attributo può essere molto utile negli elementi che comportano l'inclusione di un'immagine, dal momento che rappresenta un testo alternativo per chi non può visualizzarle. Anche un riferimento ipertestuale può avvantaggiarsi di questo attributo, perché si può visualizzare il testo corrispondente prima di raggiungere l'oggetto, in modo da avere una breve descrizione di ciò che si tratta (così da poter decidere se ne vale la pena).

```
<A HREF="http://www.brot.dg/foto/tizio.jpg"
   TITLE="Tizio in divisa">Tizio</A>
```

L'esempio mostra proprio il caso di un riferimento ipertestuale, ottenuto con l'elemento '**A**', attraverso il quale si raggiunge un file che dovrebbe mostrare l'immagine di Tizio vestito in divisa. Se il navigatore permette di conoscere il titolo del riferimento prima di doverlo raggiungere, si può evitare di prelevare il file nel caso ciò non sia interessante.

È ovvio che sta poi all'autore della pagina la scelta nello scrivere dei titoli utili o ingannevoli. Chi realizza una pagina pubblicitaria ha ovviamente degli interessi diversi da chi invece vuole realizzare un documento ordinato e facile da consultare.

54.3.5 Identificazione di un elemento

Molti elementi dispongono di un attributo '**ID**' che permette di attribuire loro un'etichetta con la quale poi farvi riferimento. Il modo tradizionale per realizzare dei riferimenti incrociati in HTML è l'uso dell'elemento '**A**', prima con l'attributo '**NAME**' (l'etichetta), poi con l'attributo '**HREF**' (il riferimento ipertestuale).

L'attributo '**ID**' permette di generalizzare il problema, dal momento che in tal modo gli elementi comuni hanno la possibilità di «identificarsi» in maniera univoca per qualunque scopo, non solo quello di definire un obiettivo per un riferimento.

```
<P ID="superparagrafo">Questo è un paragrafo nominato in
modo univoco.</P>
<P ID="supermegaparagrafo">Anche questo è un altro
paragrafo nominato in modo univoco.</P>
```

Si deve tenere presente che i nomi utilizzati per gli attributi '**ID**' devono essere univoci. Questi nomi devono essere univoci anche nei confronti dell'attributo '**NAME**' nell'elemento '**A**'.

54.3.6 Classificazione degli elementi

A differenza dell'attributo '**ID**', l'attributo '**CLASS**' consente di abbinare a un gruppo di elementi una certa classe. Il meccanismo è lo stesso, con la differenza che si vogliono indicare dei raggruppamenti. Di solito, si attribuisce una classe per abbinarne le definizioni di un foglio di stile.

```
<SPAN CLASS="nota">la vita è fatta per essere vissuta</SPAN>
```

L'esempio mostra la delimitazione di una parte di testo attraverso l'elemento '**SPAN**', al quale viene attribuita la classe '**nota**'. In seguito è possibile abbinare a tutti gli elementi di questa classe le stesse caratteristiche attraverso un foglio di stile. Utilizzando i fogli di stile CSS, si potrebbe applicare la regola seguente a tutti gli elementi '**SPAN**' della classe '**nota**':

```
SPAN.nota       { color: green; }
```

54.4 HTML: corpo

Il corpo di un documento HTML è contenuto nell'elemento '**BODY**', che può contenere blocchi di testo, intercalati da elementi '**H*n***' (da '**H1**' a '**H6**'), che rappresentano il titolo di una sezione di livello ***n***. In particolare, lo standard ISO 15445 impone che il livelli delle sezioni siano coerenti.

54.4.1 Delimitazione di blocchi e di testo normale

Per ovviare alla mancanza di una struttura prestabilita, è possibile raggruppare dei blocchi di testo o del testo normale attraverso gli elementi '**DIV**' e '**SPAN**' rispettivamente.

Gli obiettivi che ci si possono prefiggere in questo modo possono essere molti. In generale si sfrutta la possibilità di attribuire a questi elementi degli attributi per qualche scopo.

```
<DIV ID="capitolo-1" CLASS="capitolo">
<!-- contenuto del capitolo -->
...
...
</DIV>
```

Questo esempio mostra una situazione in cui l'elemento '**DIV**' viene utilizzato per delimitare una parte del corpo del documento, a cui viene attribuita la classe '**capitolo**' e la stringa di identificazione '**capitolo-1**'.

```
Il sig. <SPAN CLASS="nome">Tizio Tizi</SPAN> è andato...
```

In questo nuovo esempio, si usa l'elemento '**SPAN**' per delimitare il testo che indica il nome di una certa persona. In questo modo viene anche attribuita l'appartenenza alla classe '**nome**', cosa che può tornare utile per rendere i nomi in modo diverso attraverso un foglio di stile.

54.4.2 Titoli e struttura implicita del testo

Ciò che nel testo rappresenta un titolo di una sezione, si indica utilizzando gli elementi che vanno da '**H1**' a '**H6**'. Intuitivamente, il primo rappresenta un titolo di importanza maggiore, mentre l'ultimo è quello di importanza minore.

L'utilizzo corretto dei titoli attraverso questi elementi è molto importante perché può permettere a un sistema di visualizzazione o composizione di conoscerne la gerarchia e generare così un indice generale (se richiesto). In taluni casi si può arrivare anche a ottenere una numerazione di questi titoli in modo automatico.

```
...
<H1>Titolo principale</H1>
...
<H2>Titolo di livello inferiore</H2>
...
<H1>Altro titolo principale</H1>
...
```

Gli elementi che rappresentano dei titoli sono fatti per contenere testo normale ed elementi che non rappresentano blocchi separati.

È importante ricordare che il titolo del documento HTML, quello che si indica nell'elemento '**TITLE**' nell'intestazione, ovvero all'interno dell'elemento '**HEAD**', di norma non viene rappresentato. Per questo, spesso, il titolo del documento viene riproposto in un titolo '**H1**'.

L'esempio seguente mostra un pezzo di documento strutturato in capitoli e sezioni, delimitati formalmente attraverso l'elemento '**DIV**':

```
<H1>Trattato di bla bla bla</H1>
<DIV CLASS="capitolo" ID="capitolo-1">

<P>Questo trattato tratta di aria fritta...</P>

</DIV>
<H1>Dettagli</H1>
<DIV CLASS="sezione" ID="sezione-1-1">

<P>Questa è una sezione inutile di un trattato
inutile...</P>

</DIV>
```

Lo scopo di ciò può essere quello di attribuire stili particolari alle varie parti gerarchiche del documento. Inoltre, l'utilizzo dell'attributo '**ID**' nell'elemento '**DIV**' che introduce ogni blocco gerarchico può rappresentare un modo coerente per farvi riferimento.

È bene osservare che lo standard ISO 15445 esclude che l'elemento '**DIV**' possa contenere elementi '**H*n***'.

54.4.3 Testo

Il testo normale è ciò che è contenuto in un «blocco» di testo. Il caso più comune di blocco di testo è rappresentato dall'elemento '**P**', utilizzato per dividerlo idealmente in paragrafi.

All'interno di un blocco, salvo casi particolari, il testo viene reso in modo da adattarsi alle dimensioni imposte dal sistema di visualizzazione o di composizione. In pratica, viene suddiviso in modo conveniente, ignorando le interruzioni di riga e le righe vuote aggiunte.

È opportuno fare attenzione all'uso degli spazi all'interno degli elementi che contengono testo normale: si dovrebbe evitare di iniziare o concludere un elemento con uno spazio. In pratica, si deve evitare di scrivere qualcosa come:

```
<P>Bla bla bla <EM> evviva </EM> bla bla.</P>
```

Al suo posto bisogna invece limitarsi a scrivere:

```
<P>Bla bla bla <EM>evviva</EM> bla bla.</P>
```

54.4.3.1 Paragrafi e interruzioni

Si è già accennato al fatto che le righe vuote vengono ignorate in HTML. In effetti, l'interruzione di un paragrafo deve essere segnalata sempre esplicitamente, almeno attraverso l'indicazione dell'inizio di questo. Si osservi l'esempio seguente; anche se appare ovvio che il testo successivo alla dichiarazione del titolo è un paragrafo, questo modo non è ammissibile.

```
...
<H1>Titolo principale</H1>
Primo paragrafo che descrive qualcosa
che non serve precisare oltre.
<p>Paragrafo successivo.
<H1>Altro argomento</H1>
...
```

HTML ignora le righe bianche (possono contenere spazi e caratteri di tabulazione, oltre ai caratteri di conclusione della riga), per cui la separazione dei paragrafi attraverso l'inserzione di righe non serve a nulla.

Se si vuole ottenere l'interruzione della riga, in modo che il testo riprenda all'inizio, ma senza interrompere il paragrafo, potrebbe convenire l'utilizzo dell'elemento '**BR**', come nell'esempio seguente:

```
<P>Paragrafo che descrive qualcosa:<BR>
questa riga fa parte dello stesso paragrafo
che inizia con la riga precedente.</P>
```

Se si vogliono evitare i problemi causati dalle differenze nella composizione del risultato da parte dei programmi di navigazione, conviene ridurre al minimo l'uso di questo tipo di interruzione di riga.

Per separare il testo esiste anche la possibilità di utilizzare delle righe di separazione orizzontale: '**HR**' (*Horizontal rule*). Si tratta di elementi vuoti, per cui non si usa il marcatore di conclusione.

54.4.3.2 Elementi tipici utilizzati all'interno delle frasi

Nell'ambito del testo normale, si possono racchiudere alcune parti, per qualche motivo, all'interno di elementi specifici. Le situazioni tipiche riguardano l'evidenziamento, come nel caso degli elementi '**EM**' e '**STRONG**'.

```
... il <EM>codice di interruzione di riga</EM> è ciò
che separa le righe ...
```

La tabella 54.49 elenca gli elementi più comuni di questo tipo.

Tabella 54.49. Elementi da usare all'interno delle frasi.

Elemento	Significato
EM	Testo enfatizzato, di solito in corsivo.
STRONG	Testo evidenziato, di solito in neretto.
CITE	Citazione, nel senso di chi o cosa viene citato.
Q	Testo citato.
DFN	Definizione.
CODE	Codice usato in elaborazione, di solito reso in carattere dattilografico.
SAMP	Testo risultato di un'elaborazione.
KBD	Testo da inserire attraverso la tastiera.
VAR	Variabile o argomento di un programma.
ABBR	Abbreviazione.
ACRONYM	Acronimo.
SUB	Testo a pedice.
SUP	Testo ad apice.

Vale la pena di vedere come si può abbinare l'attributo '**TITLE**' agli elementi '**ABBR**' e '**ACRONYM**'. In generale, questi due si possono intendere quasi come la stessa cosa: spesso l'acronimo è un'abbreviazione. A parte il problema di scegliere questo o quell'elemento, l'attributo '**TITLE**' diventa utile per specificare il modo in cui si traduce l'acronimo o l'abbreviazione:

```
<ACRONYM TITLE="World Wide Web">WWW</ACRONYM>
```

A volte, un'abbreviazione o un acronimo diventano parole con un'identità propria; come tale acquisisce anche una pronuncia, che probabilmente si vuole preservare, specialmente quando il documento HTML viene letto attraverso un sistema vocale. Anche a questo scopo può essere usato l'attributo '**TITLE**'.

54.4.3.3 Citazioni

Il testo che si riferisce a una citazione si può delimitare attraverso due elementi: '**BLOCKQUOTE**' quando si tratta di blocchi di testo e '**Q**' quando si tratta di qualcosa che viene inserito nel flusso del testo normale.

```
<BLOCKQUOTE CITE="http://www.brot.dg/testi/prova.html"
 LANG="it">
<P>Bla bla bla...
bla bla bla bla...
bla bla bla.</P>
</BLOCKQUOTE>
```

Dall'esempio si può osservare l'utilizzo dell'attributo '**CITE**' con il quale si può indicare l'URI da dove ottenere il testo originale o il testo completo; inoltre è stato inserito anche l'attributo '**LANG**' allo scopo di specificare il linguaggio del testo citato (presumibilmente diverso da quello generale).

```
<P><CITE>Tizio Tizi</CITE> ha detto:
<Q CITE="http://www.brot.dg/testi/prova.html" LANG="it">
Bla bla bla...
bla bla bla bla...
bla bla bla.</Q></P>
```

Questo esempio ulteriore fa uso dell'elemento '**Q**', ma in aggiunta si vede anche l'elemento '**CITE**' con il quale viene indicato l'autore del testo citato.

54.4.3.4 Testo preformattato

In un documento HTML, l'unico modo per preservare gli spazi orizzontali e le interruzioni di riga, è l'uso dell'elemento '**PRE**'. In generale, il risultato che si ottiene viene rappresentato utilizzando un carattere dattilografico.

```
<P>Il comando <CODE>ls -l /</CODE> genera un risultato
simile a quello seguente:</P>

<PRE>
drwxr-xr-x    2 root     root         2048 gen  6 18:38 bin
drwxr-xr-x    3 root     root         1024 dic 31 08:08 boot
drwxr-xr-x    4 root     root        18432 gen 24 14:23 dev
drwxr-xr-x   68 root     root         4096 gen 24 14:09 etc
drwxr-sr-x   14 root     root         1024 gen  3 19:32 home
drwxr-xr-x    5 root     root         4096 gen  6 16:32 lib
drwxr-xr-x   19 root     root         1024 ago 15 16:02 mnt
drwxr-xr-x    5 root     root         1024 nov  9 14:59 opt
dr-xr-xr-x   88 root     root            0 gen 24 14:07 proc
drwxr-xr-x   18 root     root         1024 dic 16 17:37 root
drwxr-xr-x    3 root     root         2048 gen  6 16:12 sbin
drwxrwxrwt    6 root     root         8192 gen 24 18:56 tmp
drwxr-xr-x   16 root     root         1024 gen  5 15:23 usr
drwxr-xr-x   15 root     root         1024 set 29 15:02 var
</PRE>
```

Per essere sicuri del risultato finale, è bene evitare l'uso di caratteri di tabulazione, preferendo piuttosto gli spazi normali.

54.4.4 Elenchi

In generale, esistono tre tipi di elenchi: puntati, numerati e descrittivi. L'elenco puntato viene definito utilizzando l'elemento '**UL**' (*Unordered list*), quello numerato si ottiene con l'elemento '**OL**' (*Ordered list*), quello descrittivo si ottiene con l'elemento '**DL**' (*Definition list*). Le voci degli elenchi puntati e numerati sono costituite da elementi '**LI**' (*List item*), mentre nel caso dell'elenco descrittivo il contenuto si articola in elementi '**DT**' (*Definition term*) e '**DD**' (*Definition description*).

```
<UL>
    <LI>prima voce di un elenco puntato;</LI>
    <LI>seconda voce di un elenco puntato;</LI>
    <LI>terza voce.</LI>
</UL>
```

```
<OL>
    <LI>prima voce di un elenco numerato;</LI>
    <LI>seconda voce di un elenco numerato;</LI>
    <LI>terza voce.</LI>
</OL>
```

```
<DL>
    <DT>Resistenza</DT>
    <DT>Resistore</DT>
        <DD>Componente resistivo utilizzato in
            elettronica</DD>
    <DT>Condensatore</DT>
        <DD>Componente capacitivo...</DD>
</DL>
```

Gli esempi mostrano un uso molto semplice di questi elenchi. Si può osservare in particolare che nel caso dell'elenco descrittivo, gli elementi che delimitano il termine da descrivere possono essere più di uno simultaneamente.

Gli elementi '**LI**' e '**DT**' sono speciali, dal momento che possono contenere testo lineare normale, come si vede negli esempi, oppure dei blocchi di testo. Questo, tra le altre cose, consente di realizzare degli elenchi più complessi.

```
<OL>
    <LI><P>prima voce di un elenco numerato;</P></LI>
    <LI><P>seconda voce di un elenco numerato;</P></LI>
    <LI>
        <P>terza voce che si articola ulteriormente:</P>
        <UL>
            <LI>bla bla bla</LI>
            <LI>bla bla bla</LI>
            <LI>bla bla bla</LI>
        </UL>
    </LI>
</OL>
```

54.4.5 Tabelle

Quando si iniziano a utilizzare le tabelle e si scoprono gli effetti che si riescono a ottenere, non se ne vorrebbe più fare a meno. In realtà, sarebbe bene utilizzare le tabelle il meno possibile, perché alcuni programmi per la visualizzazione di documenti HTML non sono in grado di gestirle in maniera ottimale. Qui viene data solo una spiegazione superficiale, che comunque dovrebbe essere sufficiente per l'uso normale.

La tabella è definita dall'elemento '**TABLE**'; al suo interno può essere inclusa una didascalia rappresentata dall'elemento '**CAPTION**', quindi il contenuto della tabella viene distinto in intestazione, piede e corpo, all'interno dei quali si inseriscono le righe della tabella stessa (figura 54.58).

Figura 54.58. Esempio di una tabella.

```
.-------------------------------------------.   riga di
| Articolo       | Descrizione              |   intestazione
|================|==========================|
| 123xyz         | Bicicletta uomo          |  \
|----------------|--------------------------|  |
| 125xyz         | Bicicletta donna         |  >   corpo
|----------------|--------------------------|  |
| 121xyz         | Bicicletta bambino       |  /
|================|==========================|
| Articolo       | Descrizione              |      piede
'-------------------------------------------'
```

L'intestazione e il piede non sono obbligatori; in ogni caso, se si utilizzano vanno inseriti ordinatamente prima del corpo. Se non si indica l'intestazione o il piede, le righe che costituiscono il corpo hanno comunque bisogno di essere delimitate espressamente tra i marcatori che rappresentano l'elemento corrispondente.

Lo standard ISO 15445 obbliga all'utilizzo dell'attributo '**SUMMARY**' nell'elemento '**TABLE**'. Questo attributo dovrebbe permettere di riassumere il contenuto della tabella per quelle situazioni in cui potrebbe essere impossibile consultarla correttamente.

La tabella 54.59 riepiloga gli elementi utili nella realizzazione delle tabelle HTML.

Tabella 54.59. Elementi da usare per la realizzazione delle tabelle HTML.

Elemento	Significato
TABLE	Delimita la tabella.
CAPTION	Didascalia.
THEAD	Righe di intestazione.
TFOOT	Righe del piede.
TBODY	Righe del corpo.
TR	Riga normale.
TH	Elemento evidenziato di una riga.
TD	Elemento di una riga.

L'esempio seguente rappresenta una tabella molto banale, senza intestazione e senza piede:

```
<TABLE SUMMARY="uno due tre quattro cinque sei">
<TBODY>
    <TR><TD>uno</TD><TD>due</TD></TR>
    <TR><TD>tre</TD><TD>quattro</TD></TR>
    <TR><TD>cinque</TD><TD>sei</TD></TR>
</TBODY>
</TABLE>
```

Il risultato è uno specchietto simile a quello che si vede di seguito:

```
---------------------------------
 uno            due
 tre            quattro
 cinque         sei
---------------------------------
```

Ecco lo stesso esempio con l'aggiunta di una riga di intestazione:

```
<TABLE SUMMARY="uno due tre quattro cinque sei">
<THEAD>
    <TR><TD>Primo</TD><TD>Secondo</TD></TR>
</THEAD>
<TBODY>
    <TR><TD>uno</TD><TD>due</TD></TR>
    <TR><TD>tre</TD><TD>quattro</TD></TR>
    <TR><TD>cinque</TD><TD>sei</TD></TR>
</TBODY>
</TABLE>
```

```
---------------------------------
 Primo          Secondo
---------------------------------
 uno            due
 tre            quattro
 cinque         sei
---------------------------------
```

L'esempio seguente aggiunge anche una didascalia molto breve:

```
<TABLE SUMMARY="uno due tre quattro cinque sei">
<CAPTION>
    Tabella banale
</CAPTION>
<THEAD>
    <TR><TD>Primo</TD><TD>Secondo</TD></TR>
</THEAD>
<TBODY>
    <TR><TD>uno</TD><TD>due</TD></TR>
    <TR><TD>tre</TD><TD>quattro</TD></TR>
    <TR><TD>cinque</TD><TD>sei</TD></TR>
</TBODY>
</TABLE>
```

```
        Tabella banale
---------------------------------
 Primo          Secondo
---------------------------------
 uno            due
 tre            quattro
 cinque         sei
---------------------------------
```

Le tabelle HTML possono essere molto più complesse di quanto è stato mostrato qui. Vale la pena di sottolineare il fatto che gli elementi '**TD**', ovvero le celle all'interno delle righe, possono contenere sia testo normale, sia blocchi di testo. Inoltre, lo standard ISO 15445

non consente più l'utilizzo di attributi per la descrizione dei bordi da far risaltare, perché per questo si possono applicare degli stili.

54.4.6 Riferimenti ipertestuali

La sigla HTML fa riferimento esplicitamente a un sistema ipertestuale. Ci deve quindi essere un modo per creare questi collegamenti.

Un riferimento può essere fatto a una pagina intera o a un punto particolare di una pagina. Il riferimento può essere assoluto, cioè provvisto dell'indicazione del nodo di rete e del percorso necessario a raggiungere la pagina, oppure può essere relativo al nodo attuale.

Per i riferimenti si utilizza l'elemento '**A**' ed eventualmente l'attributo '**ID**' di molti altri elementi.

54.4.6.1 Riferimenti a una pagina intera

Un riferimento a una pagina intera, con l'indicazione del percorso assoluto per raggiungerla, viene fatto come nell'esempio seguente:

```
<A HREF="http://www.brot.dg/prove/prova.html">Pagina di
 prova</A>
```

Nell'esempio, la frase «Pagina di prova» serve come punto di riferimento del puntatore a '`http://www.brot.dg/prove/prova.html`'.

Quando di realizza un documento HTML composto da più pagine collegate tra loro, è preferibile utilizzare riferimenti relativi, in modo da non dover indicare il nome del nodo in cui si trovano e nemmeno il percorso assoluto delle directory da attraversare per raggiungerle.

```
<A HREF="varie/nota.html">Annotazioni varie</A>
```

Nell'esempio, si vede un riferimento al file '`nota.html`' contenuto nella «directory» '`varie/`' discendente dalla directory corrente. La directory corrente, in questi casi, è quella in cui si trova la pagina contenente il puntatore.[1]

Il vantaggio di utilizzare riferimenti relativi, sta nella facilità con cui il documento può essere spostato o copiato in altri punti nel file system dello stesso o di un altro elaboratore (si veda anche quanto già scritto nella sezione 54.1).

54.4.6.2 Riferimenti a una posizione di una pagina

All'interno di una pagina è possibile collocare delle etichette che poi possono servire per fare dei riferimenti, sia a partire dalla stessa pagina che da altre. L'esempio seguente mostra un esempio di un'etichetta molto semplice.

```
<A NAME="introduzione"></A>
```

Si usa quindi lo stesso elemento che serve per creare un puntatore, ma con l'attributo '**NAME**'. L'argomento dell'attributo '**NAME**' (in questo caso è la parola '**introduzione**'), identifica quel punto.

Per fare riferimento a un'etichetta nella stessa pagina si può usare la forma dell'esempio seguente, con il quale si vuole puntare all'etichetta appena creata.

```
<A HREF="#introduzione">Introduzione</A>
```

Si utilizza l'attributo '**HREF**' come al solito, ma il suo argomento è il nome dell'etichetta preceduta dal simbolo '**#**'. Evidentemente, ciò è necessario per evitare di fare riferimento a un file con lo stesso nome.

Se si vuole fare riferimento a un'etichetta di un certo file, si utilizza la notazione solita, aggiungendo l'indicazione dell'etichetta.

```
<A HREF="http://www.brot.dg/varie/linux.html#introduzione">
Introduzione a GNU/Linux</A>
```

54.4.6.3 Collegamenti simmetrici

Si può osservare che l'elemento '**A**' serve sia per indicare un'etichetta, attraverso l'attributo '**NAME**', sia per definire un riferimento, attraverso l'attributo '**HREF**' (senza contare la possibilità di usare anche l'attributo '**ID**'). Questo fatto consente di realizzare dei riferimenti simmetrici, dove un riferimento è anche etichetta della terminazione opposta:

```
<A NAME="uno" HREF="#due">vai al punto due</A>
```

```
<A NAME="due" HREF="#uno">vai al punto uno</A>
```

L'esempio dovrebbe essere abbastanza chiaro: il primo puntatore punta al secondo, che a sua volta punta al primo.

54.4.6.4 Utilizzo dell'attributo ID

L'attributo '**ID**' è una generalizzazione attraverso la quale si attribuisce un'identità a un elemento. Può essere usato come destinazione per un riferimento fatto attraverso l'elemento '**A**' con l'attributo '**HREF**', ma il suo scopo è più ampio.

In generale, quando si realizzano dei riferimenti ipertestuali dovrebbe essere più conveniente l'indicazione di etichette attraverso l'attributo '**NAME**', dal momento che ci possono essere ancora dei navigatori o altri sistemi di lettura di file HTML che non sono in grado di riconoscere l'attributo '**ID**'.

54.4.7 Inserzioni di oggetti

Un documento HTML può contenere riferimenti a «oggetti» esterni. Nei casi più comuni si tratta di immagini o di applet, ma il concetto riguarda qualunque altra cosa che possa essere incorporata nel documento. Come si può supporre, l'elemento attraverso cui si includono gli oggetti è '**OBJECT**'. La tabella 54.73 elenca alcuni degli attributi di questo elemento.

Tabella 54.73. Alcuni attributi dell'elemento '**OBJECT**'.

Attributo	Significato
DATA	Riferimento al file dell'oggetto.
TYPE	Tipo di oggetto.
STANDBY	Messaggio di attesa durante il caricamento dell'oggetto.

Come si può intuire, il minimo per importare un oggetto richiede almeno l'uso dell'attributo '**DATA**'; inoltre, in generale è opportuno aggiungere anche l'attributo '**TYPE**' per precisare subito il tipo di oggetto.

L'elemento '**OBJECT**' non può essere vuoto; ciò che racchiude è quanto deve essere mostrato nel caso non sia possibile raggiungere l'oggetto indicato, oppure non sia possibile gestire l'oggetto stesso. Di solito si tratta di testo normale, ma potrebbe trattarsi di altri oggetti alternativi.

```
<OBJECT DATA="esempio.jpg" TYPE="image/jpg">Immagine di
esempio</OBJECT>
```

L'esempio mostra l'inclusione di un'immagine, '`esempio.jpg`', che nel caso non possa essere raggiunta o visualizzata, viene rimpiazzata con la frase: «Immagine di esempio». L'esempio seguente, al contrario, tenta di visualizzare un'altra immagine in un formato alternativo; se poi anche quella non è accessibile o visualizzabile, si passa al testo di prima:

```
<OBJECT DATA="esempio.png" TYPE="image/png">
    <OBJECT DATA="esempio.jpg" TYPE="image/jpg">
        Immagine di esempio
    </OBJECT>
</OBJECT>
```

54.4.7.1 Immagini

Il tipo di immagine che può essere visualizzata dipende solo dalle limitazioni del programma di navigazione o di composizione. Generalmente si possono utilizzare solo i formati GIF, JPG e PNG (in pratica le estensioni '`.gif`', '`.jpg`' e '`.png`').[2]

I riferimenti a file di immagine si fanno attraverso l'elemento '**OBJECT**' oppure '**IMG**'. In generale, per ottenere un documento HTML adatto alla maggior parte di programmi per la navigazione, conviene ancora utilizzare il vecchio elemento '**IMG**', come nell'esempio seguente:

```
<IMG SRC="http://www.brot.dg/varie/immagini/logo.jpg"
     ALT="Logo">
```

L'elemento '**IMG**' è vuoto, pertanto non si usa il marcatore di conclusione. Come si vede dall'esempio, si utilizza l'attributo '**SRC**' per definire la collocazione del file contenente l'immagine, l'attributo '**ALT**' per indicare una descrizione alternativa nel caso in cui l'immagine non possa essere visualizzata. La stessa cosa avrebbe potuto essere espressa con l'elemento '**OBJECT**' nel modo seguente:

```
<OBJECT DATA="http://www.brot.dg/varie/immagini/logo.jpg"
        TYPE="image/jpg">Logo</OBJECT>
```

Generalmente, per evitare problemi di compatibilità con i vari programmi di navigazione, è meglio evitare di fare scorrere il testo a fianco delle immagini, per cui è bene staccare il testo normale racchiudendolo esplicitamente all'interno di un elemento '**P**' (paragrafo).

```
<IMG SRC="immagini/logo.jpg" ALT="Logo">
<P>...testo che segue l'immagine...
```

L'immagine può essere utilizzata anche come pulsante per un riferimento ipertestuale, quando è contenuta all'interno di questo ultimo. In tali casi è particolarmente importante ricordare di inserire l'attributo '**ALT**', che diventa un'alternativa indispensabile nel caso in cui l'immagine non possa essere visualizzata.

```
<A HREF="varie/nota.html"><IMG SRC="img/nota.jpg"
ALT="Annotazioni varie"></A>
```

Naturalmente, se fosse necessario ricordarlo, non è obbligatorio che tutto si trovi sulla stessa riga, quindi l'esempio precedente può anche essere assemblato come indicato qui sotto:

```
<A HREF="varie/nota.html">
    <IMG SRC="immagini/nota.jpg" ALT="Annotazioni varie">
</A>
```

54.5 XHTML

XHTML è una rivisitazione dell'HTML in forma di applicazione XML, rimanendo molto simile all'HTML tradizionale, ma con alcune differenze importanti. In particolare:

- gli elementi devono essere delimitati correttamente con i marcatori di apertura e chiusura;
- non ci possono più essere elementi vuoti indicati con il solo marcatore di apertura, dal momento che al loro posto si possono solo usare i marcatori speciali nella forma '**<...>**';[3]
- i nomi degli elementi e degli attributi vanno scritti utilizzando solo lettere minuscole;
- gli attributi devono essere assegnati correttamente (non si possono usare più degli attributi booleani) e il valore assegnato deve essere delimitato da apici doppi o singoli;
- l'attributo '**lang**', se utilizzato, deve essere abbinato anche all'attributo '**xml:lang**', in base alle convenzioni dell'XML;
- se non si può evitare l'uso dell'attributo '**name**', questo deve essere abbinato anche all'attributo '**id**';
- se il valore assegnato a un attributo deve contenere una e-commerciale ('**&**'), occorre indicarla nella forma '**&**', anche se si tratta di un URI;
- se per qualche ragione non si dichiara la codifica utilizzata, deve trattarsi della forma UTF-8 oppure UTF-16;
- l'elemento '**isindex**' è superato e si preferisce usare l'elemento '**input**'.

54.5.1 Scheletro di un file XHTML

Trattandosi di un'applicazione XML, l'inizio dovrebbe essere scontato: si deve specificare che si tratta di un file XML, quindi si passa a indicare il DTD a cui si fa riferimento:


```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="it"
      lang="it">
<head>
    <title>Esempio XHTML</title>
    <meta name="keywords" content="XML, SGML, XSL, DSSSL" />
</head>
<body>
    <p>Ciao mondo!</p>
</body>
</html>
```

L'esempio mostra un file XHTML completo, anche se molto breve. Si può osservare che il marcatore di apertura, oltre agli attributi '**xml:lang**' e '**lang**', contiene l'attributo '**xmlns**', a cui viene assegnato un URI prestabilito.

In XML, l'insieme di caratteri codificato è quello dell'insieme di caratteri universale. Di conseguenza, per la migliore compatibilità con il passato, la forma codificata del carattere più appropriata è UTF-8. Se il file utilizza l'ASCII tradizionale, senza estensioni, tutto va bene e non occorre altro; diversamente vanno usate preferibilmente le codifiche UTF-8 oppure UTF-16, come prevede in generale l'XML. La forma codificata del carattere viene specificata nell'istruzione iniziale, come già mostrato nell'esempio iniziale:

```
<?xml version="1.0" encoding="UTF-8" ?>
```

54.5.2 Verifica della validità di un file XHTML

Volendo verificare la validità di un file XHTML attraverso il suo DTD, si può agire in modo simile a quanto si fa in generale con l'SGML. Questo è già descritto nella sezione 51.2; tuttavia occorre ricordare che la definizione SGML da utilizzare è quella specifica per l'XML.

Il DTD di XHTML e la definizione delle entità standard, possono essere ottenuti a partire da '`http://www.w3.org/TR/xhtml1/DTD/`' (ma probabilmente la directory non è leggibile), mentre la dichiarazione SGML si può trovare tra i sorgenti del pacchetto SP di James Clark. Si veda a questo proposito quanto già descritto nel capitolo 52.

54.6 CSS

I fogli di stile CSS (*Cascading style sheet*) rappresentano un metodo semplice per consentire di dichiarare e abbinare degli stili di composizione ai documenti HTML e ad altri tipi di sistemi SGML.

Il lavoro su CSS ha generato due «livelli», CSS1 e CSS2, intesi come la prima e la seconda versione del linguaggio di stile CSS. Teoricamente, il linguaggio CSS deve essere compatibile sia verso l'alto, sia verso il basso, nel senso che il primo livello CSS è compatibile con il secondo e il secondo è compatibile con il primo. In pratica, le estensioni fatte al linguaggio nel CSS2 sono tali per cui dovrebbero essere ignorate semplicemente dai programmi in grado di interpretare correttamente solo CSS1.

Nelle sezioni successive si introduce il linguaggio CSS affrontando solo parte delle caratteristiche del primo livello, con qualche annotazione eventuale sul secondo. Nella sezione 54.2.2 è descritto in quanti modi si può includere un foglio di stile CSS in un documento HTML, pertanto questo particolare non viene riproposto.

54.6.1 Logica del linguaggio CSS

Nella documentazione di CSS, le sue istruzioni vengono definite «regole», che si esprimono sinteticamente secondo la forma seguente, dove le parentesi graffe fanno parte della dichiarazione della regola:

selettore { ***dichiarazione*** }

Il principio è molto semplice: il «selettore» rappresenta qualcosa all'interno del documento; la dichiarazione è ciò che si vuole ottenere

su tale oggetto. All'interno di una regola si possono raggruppare più selettori, applicando così le stesse dichiarazioni; inoltre si possono indicare più dichiarazioni: i selettori si separano con la virgola; le dichiarazioni si separano con un punto e virgola:

```
selettore [, selettore]... { dichiarazione [; dichiarazione]... }
```

Le regole possono essere scritte anche utilizzando più righe di testo normale, per cui, la stessa sintassi appena mostrata potrebbe essere scritta anche come nel modo seguente (si osservi l'aggiunta di un punto e virgola ulteriore):

```
selettore [, selettore]... {
    dichiarazione ;
    [dichiarazione ;]
    ...
}
```

Teoricamente, quando si scrivono le regole iniziando ogni dichiarazione in una riga separata, è possibile evitare l'aggiunta del punto e virgola finale, ma questa scorciatoia non è consigliabile in generale.

Le dichiarazioni si scompongono a loro volta in proprietà e valori loro assegnati:

```
selettore [, selettore]... {
    proprietà : valore [valore_alternativo]... ;
    [proprietà : valore [valore_alternativo]... ;]
    ...
}
```

Come si vede, alle proprietà si possono assegnare più valori alternativi, in ordine di importanza.

Si osservi l'esempio seguente: attribuisce il colore blu al testo degli elementi '**H1**' di un documento HTML:

```
H1 { color: blue }
```

L'esempio successivo indica l'utilizzo di uno sfondo composto da un'immagine esterna per il corpo del documento, specificando che in mancanza dell'immagine, o in mancanza della possibilità di rappresentarla si può utilizzare uno sfondo bianco:

```
BODY { background: url(fondale.jpg) white }
```

Si intuisce che il nome del file contenente l'immagine è stato indicato come argomento di quello che sembra essere una funzione: ***url()***. Si osservi comunque che questa funzione fa riferimento a un URI e non a un URL, come fa intendere erroneamente il suo nome.

I commenti in un foglio di stile CSS si rappresentano in modo simile al linguaggio C, nella forma:

```
/* testo_ignorato */
```

54.6.1.1 Ereditarietà e collegamento in cascata

Una caratteristica fondamentale del linguaggio CSS è l'ereditarietà di talune caratteristiche in certe circostanze. Per comprendere il significato della cosa basta pensare alla struttura dell'HTML, o a un altro linguaggio SGML in generale: se si attribuisce una caratteristica stilistica a un elemento che per sua natura ne può contenere altri, ci si aspetta intuitivamente che questa si trasmetta anche ai livelli inferiori se applicabile, a meno che per tali elementi sia stato definito espressamente qualcosa di diverso.

Volendo fare un esempio più pratico, si può immaginare una caratteristica riferita alla dimensione del carattere di un blocco di testo. Se questo blocco contiene delle porzioni di testo delimitate da altri elementi, che possono servire per ottenere un testo enfatizzato in qualche modo, è normale attendersi che per queste porzioni venga utilizzata la stessa dimensione del carattere, senza bisogno di dichiarare esplicitamente e dettagliatamente questa richiesta.[4]

In generale, per quanto riguarda l'HTML, è normale assegnare all'elemento '**BODY**' le caratteristiche generali di tutto il documento, sfruttando il principio di ereditarietà.

L'altra caratteristica fondamentale del linguaggio CSS è la possibilità di definire gli stili in cascata. Questo significa che si possono abbinare assieme più fogli di stile e che nel complesso ci possono essere regole che si contraddicono a vicenda. Evidentemente, in questi casi viene applicato un criterio di scelta, che viene descritto più avanti.

54.6.1.2 Selettori

Il selettore di una regola CSS è qualcosa che rappresenta una parte del testo a cui si vogliono applicare le dichiarazioni relative. Nella situazione più semplice, il selettore viene indicato con il nome dell'elemento a cui si attribuisce. In questo modo, le dichiarazioni si applicano a tutti gli elementi di quel tipo. Nell'esempio seguente, che è già stato usato in precedenza, si attribuisce il colore blu al testo che compone tutti gli elementi '**H1**':

```
H1 { color: blue }
```

Tutti gli elementi HTML che si possono utilizzare nel corpo di tale tipo di documento possono utilizzare l'attributo '**CLASS**'. Questo permette di attribuire loro una «classe», ovvero un gruppo, di solito nell'ambito di quel tipo di elemento. Per indicare un selettore che faccia riferimento a una classe specifica di un certo elemento, si usa la notazione seguente:

```
[elemento] . classe
```

Come si vede, l'indicazione dell'elemento è facoltativa, in modo tale che, se non lo si indica, si faccia riferimento a tutti gli elementi che appartengono a quella stessa classe. L'esempio seguente mostra il caso degli elementi '**P**' che appartengono alla classe '**nota**', a cui viene abbinato il colore rosso per il testo:

```
P.nota { color: red }
```

L'esempio seguente mostra invece l'utilizzo di un selettore che fa riferimento a una classe di qualunque elemento:

```
.calmante { color: green }
```

Un selettore può essere anche più specifico e arrivare a individuare un elemento preciso nel documento HTML, attraverso il riferimento all'attributo '**ID**':

```
[elemento]#identificazione
```

In questa situazione non è necessario indicare il nome dell'elemento, dato che la stringa di identificazione è già un dato univoco per conto proprio. Al contrario, se si sbaglia l'indicazione dell'elemento, si annulla la validità della regola relativa, perché non può essere applicata. L'esempio seguente attribuisce all'elemento '**P**' identificato dalla stringa '**xyz**' il colore blu:

```
P#xyz { color: blu }
```

Un selettore può essere composto in modo da definire la dipendenza da un contesto. In altri termini, si può definire un selettore che dipende da un altro:

```
selettore sottoselettore [sotto_sottoselettore]...
```

Il primo selettore indica un ambito, all'interno del quale va cercata la corrispondenza per il secondo selettore, continuando eventualmente ad aumentare il dettaglio con altri selettori più specifici. Si osservi l'esempio seguente; serve a fare riferimento agli elementi '**EM**' che si trovano all'interno di un elemento '**H1**':

```
H1 EM { color: green }
```

È importante distinguere il raggruppamento di selettori dalla definizione di un contesto più dettagliato come in questo caso. Infatti, per

raggruppare i selettori si utilizza la virgola. L'esempio seguente applica il colore verde a tutti gli elementi '**EM**' contenuti all'interno di elementi '**H1**' o '**H2**':

```
H1 EM, H2 EM { color: green }
```

Un selettore può anche individuare una pseudo-classe, ovvero una zona di testo che viene individuata dal programma che si occupa di interpretare il documento HTML, che non corrisponde a elementi e classi indicati espressamente:

```
[elemento][.classe]:pseudo_classe
```

Il caso tipico di una pseudo-classe è quella che delimita la prima lettera di un elemento: '**first-letter**'. L'esempio seguente serve a ottenere una lettera iniziale più grande in tutti gli elementi '**P**' di classe '**primo**':

```
P.primo:first-letter {
    font-size: 200%;
    float: left;
}
```

54.6.1.3 Stili in cascata

I fogli di stile CSS possono essere uniti assieme in cascata. Tra le altre cose, ciò permette la definizione di uno o più stili da parte dell'autore e di uno o più stili personalizzati da parte dell'utente che legge il documento. Un file contenente lo stile CSS può incorporare altri file attraverso la direttiva '**@import**' che ha la sintassi seguente:

```
@import url(uri_foglio_di_stile);
```

Come si vede, riappare la funzione ***url()*** già mostrata in precedenza. In generale, le direttive di incorporazione dei fogli di stile esterni vanno collocate all'inizio del file, prima delle regole CSS.

Si è accennato al fatto che, nell'ambito dello stile complessivo che si ottiene, si possono generare dei conflitti tra dichiarazioni riferite alla stessa porzione di documento. Per scegliere quale dichiarazione deve avere la meglio, è necessario stabilire un peso differente, che dipende dal contesto e può anche essere condizionato attraverso l'aggiunta della stringa '**! important**' in coda alla dichiarazione:

```
H1 {
    color: black ! important;
    background: white ! important;
}
```

L'esempio mostra il caso in cui si tenta di aumentare il peso delle dichiarazioni che definiscono il colore del testo e dello sfondo negli elementi '**H1**'.

Viene descritta brevemente e in modo semplificato la sequenza attraverso cui vengono attribuite le caratteristiche dello stile.

- Le dichiarazioni vengono applicate se c'è la corrispondenza con i selettori. Se non ci sono corrispondenze, si applicano i valori ereditati; se non è possibile ereditare alcunché, si usano i valori iniziali.
- Le dichiarazioni vengono ordinate in base al loro peso, dove quelle marcate come «importanti» ricevono un peso maggiore rispetto a quelle normali.
- Le dichiarazioni vengono ordinate in base alla loro origine: lo stile dell'autore ha la precedenza su quello personalizzato dell'utente, che a sua volta ha la precedenza su quello predefinito dal programma utilizzato.
- Le dichiarazioni vengono ordinate in base alla precisione con cui individuano gli obiettivi. In pratica, le dichiarazioni più specifiche hanno la precedenza rispetto a quelle più generali.
- Al termine, se due regole hanno lo stesso peso, ha la precedenza quella che appare per ultima.

54.6.2 Proprietà

Di seguito vengono mostrate una serie di tabelle che descrivono l'utilizzo di alcune proprietà comuni nel linguaggio CSS. Bisogna ricordare che ogni programma di lettura o di composizione dei documenti HTML può fare la propria scelta su quali siano le dichiarazioni da prendere in considerazione, ignorando tutto il resto. Pertanto, si tratta solo di un'indicazione e l'utilizzo degli stili CSS deve essere sempre valutato tenendo conto delle carenze che poi ci possono essere in fase di lettura.

Tabella 54.93. Proprietà riferite ai caratteri.

Proprietà	Valori	Descrizione
`font-family`	*tipo_di_carattere*	Tipo di carattere.
`font-style`	`normal`	Forma normale.
	`italic`	Corsivo.
	`oblique`	Obliquo.
`font-variant`	`normal`	Serie normale.
	`small-caps`	Maiuscoletto.
`font-weight`	normal	Tono normale.
	`bold`	Nero.
	`bolder`	Nerissimo.
	`lighter`	Chiaro.
`font-size`	*n*`pt`	Dimensione in punti.
	n`cm`	Dimensione in centimetri.
	n`mm`	Dimensione in millimetri.
	n`em`	Dimensione relativa in quadratoni.
	n`ex`	Dimensione relativa in Ex.
	n`%`	Dimensione relativa percentuale.
	`small`	Carattere piccolo.
	`medium`	Carattere normale.
	`large`	Carattere grande.

Nella tabella 54.93 si fa riferimento in particolare alla proprietà '**font-family**'. A questa può essere attribuito il nome di una famiglia di caratteri, oppure il nome di una «famiglia generica», che in pratica identifica uno stile del carattere senza indicare esattamente quale tipo di carattere. Una famiglia di caratteri potrebbe essere '**times**', mentre una famiglia generica potrebbe essere '**serif**', ovvero un carattere munito di grazie. Alla proprietà '**font-family**' possono essere abbinati più tipi di caratteri, separati da una virgola, per indicare una sequenza alternativa da utilizzare in mancanza di quello preferito:

```
BODY { font-family: gill, helvetica, sans-serif }
```

L'esempio mostra proprio questo: prima si tenta di utilizzare il carattere '**gill**'; quindi si prova con '**helvetica**'; infine ci si accontenta di un carattere senza grazie, '**sans-serif**'.

Tabella 54.95. Proprietà riferite ai colori e allo sfondo.

Proprietà	Valori	Descrizione
`color`	*colore*	Colore del carattere o di primo piano.
`background-color`	*colore*	Colore dello sfondo.
`background-image`	`url(`*uri*`)`	Immagine da usare per lo sfondo.

Per quanto riguarda i colori (tabella 54.95), si possono indicare attraverso il nome che questi hanno in inglese, oppure attraverso la funzione ***rgb()***, con la quale si specifica il valore RGB:

```
rgb(livello_rosso, livello_verde, livello_blu)
```

I numeri che esprimono i livelli dei colori fondamentali RGB vanno da 0 a 255.

Tabella 54.96. Proprietà riferite al testo.

Proprietà	Valori	Descrizione
`vertical-align`	`baseline`	Testo al livello normale.
	`middle`	Allinea al centro.
	`sub`	Pedice.
	`super`	Apice.
`text-transform`	`none`	Nessuna trasformazione del testo.
	`capitalize`	Rende maiuscola la prima lettera delle parole.
	`uppercase`	Tutto maiuscolo.
	`lowercase`	Tutto minuscolo.
`text-align`	`left`	Allinea a sinistra.
	`right`	Allinea a destra.
	`center`	Centra.
	`justify`	Allinea a sinistra e a destra.
`text-indent`	*n*`pt`	Rientro in punti.
	n`cm`	Rientro in centimetri.
	n`mm`	Rientro in millimetri.
	n`em`	Rientro relativo in quadratoni.
	n`ex`	Rientro relativo in Ex.
	n`%`	Rientro relativo in percentuale.
`line-height`	`normal`	Altezza normale della riga.
	n`pt`	Altezza in punti.
	n`cm`	Altezza in centimetri.
	n`mm`	Altezza in millimetri.
	n`%`	Altezza relativa in percentuale.

Tabella 54.97. Proprietà riferite al testo racchiuso in blocchi rettangolari.

Proprietà	Valori	Descrizione
`margin-top`	`auto`	Margine superiore automatico.
	n`pt`	Margine superiore in punti.
	n`cm`	Margine superiore in centimetri.
	n`mm`	Margine superiore in millimetri.
	n`%`	Margine superiore relativo in percentuale.
`margin-bottom`	`auto`	Margine inferiore automatico.
	n`pt`	Margine inferiore in punti.
	n`cm`	Margine inferiore in centimetri.
	n`mm`	Margine inferiore in millimetri.
	n`%`	Margine inferiore relativo in percentuale.
`margin-left`	`auto`	Margine sinistro automatico.
	n`pt`	Margine sinistro in punti.
	n`cm`	Margine sinistro in centimetri.
	n`mm`	Margine sinistro in millimetri.
	n`%`	Margine sinistro relativo in percentuale.
`margin-right`	`auto`	Margine destro automatico.
	n`pt`	Margine destro in punti.
	n`cm`	Margine destro in centimetri.
	n`mm`	Margine destro in millimetri.
	n`%`	Margine destro relativo in percentuale.
`border-width`	`thin`	Bordo sottile.
	`medium`	Bordo medio.
	`thick`	Bordo spesso.
`border-color`	***colore***	Colore del bordo.
`border-style`	`none`	Bordo non visibile.
	`dotted`	Bordo puntato.
	`dashed`	Bordo tratteggiato.
	`solid`	Bordo continuo.
	`double`	Bordo continuo doppio.
`width`	`auto`	Larghezza automatica.
	n`pt`	Larghezza in punti.
	n`cm`	Larghezza in centimetri.
	n`mm`	Larghezza in millimetri.
	n`%`	Larghezza relativa in percentuale.
`height`	`auto`	Altezza automatica.
	n`pt`	Altezza in punti.
	n`cm`	Altezza in centimetri.
	n`mm`	Altezza in millimetri.
	n`%`	Altezza relativa in percentuale.
`float`	`none`	Posizione fissa.
	`left`	A sinistra con testo che scorre a destra.
	`right`	A destra con testo che scorre a sinistra.
`clear`	`none`	Scorre normalmente.
	`left`	Salta un oggetto che si trova a sinistra.
	`right`	Salta un oggetto che si trova a destra.
	`both`	Salta qualunque oggetto fluttuante.

54.6.3 Definizione della pagina

Il secondo livello del linguaggio CSS, introduce una regola speciale, '**@page**', per la definizione della pagina, nel momento in cui il documento dovesse essere stampato. Inoltre, sono disponibili delle proprietà specifiche per l'impaginazione da usarsi nelle regole normali. In generale, la regola '**@page**' viene usata per definire i margini ed eventualmente anche le dimensioni della pagina. L'esempio seguente dichiara una pagina A4 utilizzando margini tutti uguali di 2 cm:

```
@page {
  size 210mm 297mm;
  margin-top: 2cm;
  margin-bottom: 2cm;
  margin-left: 2cm;
  margin-right: 2cm;
}
```

La stessa cosa si potrebbe ottenere in modo meno dettagliato come segue:

```
@page {
  size 210mm 297mm;
  margin: 2cm;
}
```

La tabella 54.100 riepiloga le proprietà più importanti riferite a questa regola.

Tabella 54.100. Proprietà riferite alla regola speciale '**@page**'.

Proprietà	Valori	Descrizione
`size`	*x y*	Ampiezza e altezza della pagina (nelle varie unità di misura).
	`auto`	Definisce le dimensioni e l'orientamento in modo automatico.
	`landscape`	Orientamento orizzontale.
	`portrait`	Orientamento verticale.
`margin`	*x*	Dimensione di tutti i margini.
	n`pt`	Dimensione in punti.
	n`cm`	Dimensione in centimetri.
	n`mm`	Dimensione in millimetri.
	n`%`	Dimensione relativa in percentuale.
`margin-left`	*x*	Dimensione del margine sinistro.
`margin-right`	*x*	Dimensione del margine destro.
`margin-top`	*x*	Dimensione superiore.
`margin-bottom`	*x*	Dimensione inferiore.

La regola '**@page**' può essere usata in modo da distinguere tra pagine destre e pagine sinistre. Si osservi a questo proposito l'esempio seguente:

```
@page :left {
  margin-top: 2cm;
  margin-bottom: 2cm;
  margin-left: 4cm;
  margin-right: 2cm;
}

@page :right {
  margin-top: 2cm;
  margin-bottom: 2cm;
  margin-left: 2cm;
  margin-right: 4cm;
}
```

Come accennato sono disponibili delle proprietà specifiche per l'impaginazione da usarsi nelle regole normali. Con queste si intende controllare la suddivisione del testo in pagine, imponendo un salto pagina, oppure impedendolo nell'ambito dell'elemento coinvolto.

Queste proprietà non vengono descritte qui, ma è utile almeno tenere in considerazione la loro esistenza.

54.7 JavaScript

Il linguaggio JavaScript somiglia al Java, semplificato in modo da poter essere eseguito direttamente, come uno script comune. Questo linguaggio è però destinato a essere interpretato da un navigatore, inserito normalmente all'interno di documenti HTML, pertanto il suo ambito di utilizzo è più limitato rispetto a Java.

Oltre alla limitatezza del contesto a cui è destinato questo linguaggio di programmazione, occorre considerare il fatto che, essendo interpretato dal navigatore, può esistere solo una compatibilità di massima, perché tutto dipende dalla capacità del navigatore stesso di eseguire le istruzioni del linguaggio.

Si intende che il ruolo del linguaggio JavaScript è quello di arricchire di potenzialità i documenti ipertestuali che si possono leggere attraverso un navigatore, ma in generale il suo utilizzo va ponderato, per non escludere quella porzione di utenti che si trova a usare software che per qualche ragione non può eseguire script di questo tipo.

Prima di poter affrontare la descrizione del linguaggio, è necessario comprendere come si usa in pratica, dal momento che ciò richiede l'inserimento, in qualche modo, all'interno di documenti ipertestuali. Si osservi l'esempio seguente che mostra il contenuto di un file HTML molto semplice:

```
<!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
<HTML LANG="it">
<HEAD>
    <META HTTP-EQUIV="Content-Type" CONTENT="text/html">
    <TITLE>JavaScript 1</TITLE>
    <SCRIPT LANGUAGE="JavaScript" TYPE="text/javascript">
        <!--
           document.write ("Ciao a tutti! ");
           document.write ("Questo &egrave; il mio primo ");
           document.write ("programma JavaScript.");
        //-->
    </SCRIPT>
</HEAD>
<BODY>
<H1>JavaScript 1</H1>
<P>Bla bla bla bla...</P>
</BODY>
</HTML>
```

Se si legge questo file con un navigatore comune, si dovrebbe ottenere di visualizzare questo risultato:

Ciao a tutti! Questo è il mio primo programma JavaScript.

JavaScript 1

Bla bla bla bla...

Per cominciare si deve osservare che è stato utilizzato l'elemento '**SCRIPT**' all'interno dell'intestazione del file. Attraverso l'attributo '**LANGUAGE**' è stato specificato l'uso del linguaggio JavaScript e con l'attributo '**TYPE**' (che in questo caso è facoltativo) viene specificato il tipo MIME.

Il contenuto dell'elemento '**SCRIPT**' è il programma JavaScript che, come si vede dal risultato, viene eseguito prima di analizzare il corpo della pagina HTML. Per garantire di non ottenere effetti spiacevoli in presenza di un navigatore che non conosce l'uso di questo elemento, convenzionalmente, si delimita il codice inserito attraverso un commento particolare:

```
    <SCRIPT LANGUAGE="JavaScript" TYPE="text/javascript">
        <!--
            …
            …
```

```
            ...
        //-->
    </SCRIPT>
```

Si osservi che la chiusura del commento deve essere realizzata necessariamente secondo la forma '`//-->`', per garantire la massima compatibilità con i navigatori.

In alternativa, si può ottenere lo stesso risultato scrivendo il codice JavaScript in un file separato:

```
document.write ("Ciao a tutti! ");
document.write ("Questo &egrave; il mio primo ");
document.write ("programma JavaScript.");
```

Supponendo che questo file si chiami '`javascript-001.js`', il file HTML, collocato nella stessa directory, potrebbe essere realizzato semplicemente così:

```
<!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
<HTML LANG="it">
<HEAD>
    <META HTTP-EQUIV="Content-Type" CONTENT="text/html">
    <TITLE>JavaScript 1</TITLE>
    <SCRIPT LANGUAGE="JavaScript" TYPE="text/javascript"
        SRC="javascript-001.js"></SCRIPT>
</HEAD>
<BODY>
<H1>JavaScript 1</H1>
<P>Bla bla bla bla...</P>
</BODY>
</HTML>
```

Il risultato che si ottiene è lo stesso.

Negli esempi che vengono mostrati qui a proposito delle pagine HTML che incorporano o fanno riferimento a programmi JavaScript, si specifica in modo esplicito il tipo di standard HTML usato e si cerca di scrivere in modo corretto, secondo la sintassi prevista. Tuttavia, nessuno degli standard HTML, tanto meno lo standard ISO che viene usato proprio negli esempi, prevede l'inserzione o il riferimento a programmi JavaScript nel modo mostrato. Pertanto, questi file sono errati formalmente, per ciò che riguarda il linguaggio HTML, ma rappresentano l'unico modo per garantire un discreto grado di compatibilità tra i navigatori che devono eseguire gli script contenuti.

Negli esempi mostrati appare un solo elemento '**`SCRIPT`**' nell'intestazione del file HTML. In pratica ci possono essere più elementi '**`SCRIPT`**' nell'intestazione e ne possono apparire anche nel corpo del documento, come nell'esempio seguente:

```
<!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
<HTML LANG="it">
<HEAD>
    <META HTTP-EQUIV="Content-Type" CONTENT="text/html">
    <TITLE>JavaScript 2</TITLE>
    <SCRIPT LANGUAGE="JavaScript" TYPE="text/javascript">
        <!--
            document.write ("Ciao a tutti! ");
        //-->
    </SCRIPT>
    <SCRIPT LANGUAGE="JavaScript" TYPE="text/javascript">
        <!--
            document.write ("Questo &egrave; il mio ");
            document.write ("secondo programma ");
            document.write ("JavaScript.");
        //-->
    </SCRIPT>
</HEAD>
<BODY>
<H1>JavaScript 2</H1>
<P>Bla bla bla bla...</P>

<SCRIPT LANGUAGE="JavaScript" TYPE="text/javascript">
    <!--
```



```
        document.write ("Come si pu&ograve; vedere, ");
        document.write ("&egrave; possibile inserire ");
        document.write ("del codice JavaScript ");
        document.write ("anche nel corpo del file HTML.");
    //-->
</SCRIPT>
</BODY>
</HTML>
```

Ecco il risultato che si dovrebbe ottenere:

Ciao a tutti! Questo è il mio secondo programma JavaScript.

JavaScript 2

Bla bla bla bla...

Come si può vedere, è possibile inserire del codice JavaScript anche nel corpo del file HTML.

Gli esempi mostrati fino a questo punto fanno uso esclusivo della funzione ***document.write()***, ovvero, più precisamente, del metodo ***write()*** dell'oggetto '**`document`**'. Ciò che si ottiene utilizzandolo è di inserire la stringa che viene data come argomento nel documento che viene visualizzato. Pertanto, quanto viene inserito nell'intestazione appare all'inizio, mentre ciò che viene inserito nel corpo, appare in corrispondenza dell'inserzione.

Dal momento che il documento in questione è scritto con il linguaggio HTML, queste stringhe devono essere coerenti con il linguaggio HTML stesso, cosa che negli esempi mostrati non è stato, a parte l'uso di alcune macro per le vocali accentate. L'esempio appena presentato, andrebbe modificato, come appare qui di seguito:

```
<!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
<HTML LANG="it">
<HEAD>
    <META HTTP-EQUIV="Content-Type" CONTENT="text/html">
    <TITLE>JavaScript 3</TITLE>
    <SCRIPT LANGUAGE="JavaScript" TYPE="text/javascript">
        <!--
            document.write ("<P>Ciao a tutti! ");
        //-->
    </SCRIPT>
    <SCRIPT LANGUAGE="JavaScript" TYPE="text/javascript">
        <!--
            document.write ("Questo &egrave; il mio ");
            document.write ("secondo programma ");
            document.write ("JavaScript.</P>");
        //-->
    </SCRIPT>
</HEAD>
<BODY>
<H1>JavaScript 3</H1>
<P>Bla bla bla bla...</P>

<SCRIPT LANGUAGE="JavaScript" TYPE="text/javascript">
    <!--
        document.write ("<P>Come si pu&ograve; vedere, ");
        document.write ("&egrave; possibile inserire del ");
        document.write ("codice JavaScript anche nel ");
        document.write ("corpo del file HTML.</P>");
    //-->
</SCRIPT>
</BODY>
</HTML>
```

In pratica, viene circoscritto il testo all'interno di un elemento '**`P`**'. Naturalmente, nello stesso modo si può inserire del codice HTML più complesso, anche se per evitare di fare troppa confusione, sarebbe meglio ridurre al minimo questa possibilità.

Generalmente, quando si scrive un programma JavaScript non si può disporre di un analizzatore ortografico per la ricerca di errori, anche se banali. In tal modo, l'unica cosa su cui si può contare è il programma che lo interpreta, ovvero il navigatore, che normalmente si limita a non eseguire il programma che non sia sintatticamente perfetto. Pertanto, ciò significa che la scrittura di programmi complessi diventa un compito molto difficile e a volte impossibile.

54.7.1 Verifica sintattica

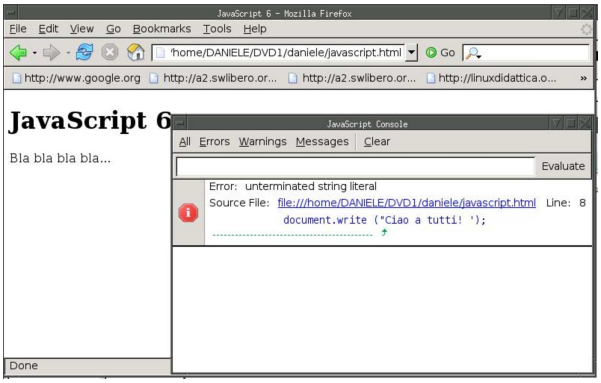
Al programmatore JavaScript manca un analizzatore sintattico standardizzato universale e il controllo si può eseguire solo attraverso gli strumenti di questo o di quel navigatore. Fortunatamente, Mozilla (Netscape) e molti dei suoi vari derivati, offre la «console JavaScript», dove vengono messi in evidenza quelli che possono essere degli errori del programma.

Viene proposto un esempio HTML contenente un programma JavaScript con diversi errori.

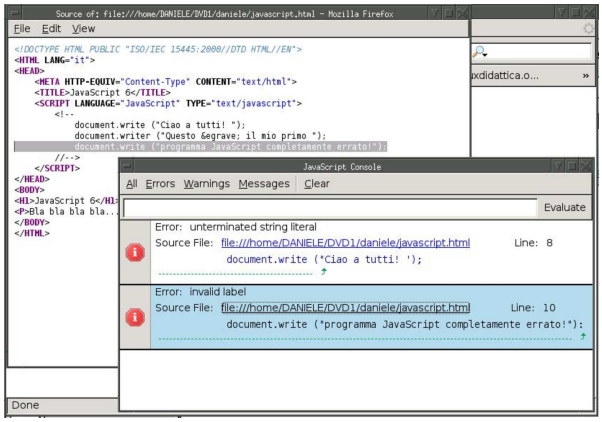
Inizialmente viene visualizzato il file con Mozilla Firefox e viene aperta la console JavaScript selezionando la voce *JavaScript Console* dal menù *Tools*.

```
<!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
<HTML LANG="it">
<HEAD>
    <META HTTP-EQUIV="Content-Type" CONTENT="text/html">
    <TITLE>JavaScript 6</TITLE>
    <SCRIPT LANGUAGE="JavaScript" TYPE="text/javascript">
        <!--
            document.write ("Ciao a tutti! ');
            document.writer ("Questo &egrave; il mio ");
            document.writer ("primo programma JavaScript ");
            document.write ("completamente errato!"):
        //-->
    </SCRIPT>
</HEAD>
<BODY>
<H1>JavaScript 6</H1>
<P>Bla bla bla bla...</P>
</BODY>
</HTML>
```

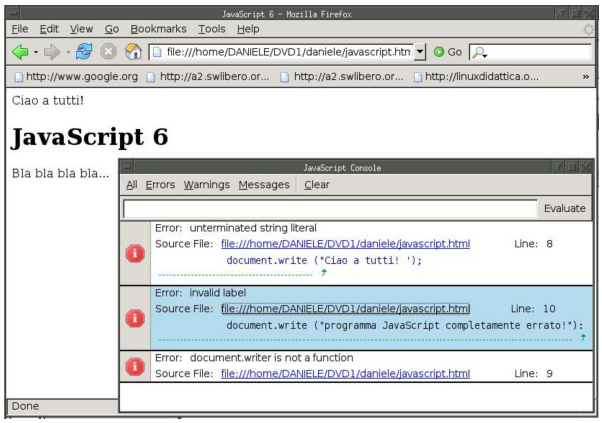
Inizialmente viene visualizzato il file con Mozilla Firefox e viene aperta la console JavaScript selezionando la voce *JavaScript Console* dal menù *Tools*:



Come si vede dalla figura, il primo errore a essere segnalato è la stringa terminata in modo errato. Si ipotizza di correggere l'errore e di ricaricare la pagina:



In questo caso, seguendo il collegamento visibile in corrispondenza dell'errore che viene segnalato, si vede apparire una finestra contenente il sorgente con la riga errata evidenziata. L'errore che viene messo in risalto è dovuto alla conclusione dell'istruzione con i due punti, invece che con il punto e virgola; si ipotizza di correggere anche questo errore e di ricaricare la pagina:



L'ultimo errore a essere notato è il nome di una funzione inesistente: '`writer`'.

Altri navigatori derivati da Mozilla offrono una console simile, eventualmente da voci diverse del menù.

54.7.2 Caratteristiche generali del linguaggio di programmazione

Il linguaggio JavaScript è molto vicino a Java, con delle semplificazioni significative. Le istruzioni hanno la forma del linguaggio C, pertanto terminano con un punto e virgola ('`;`'); inoltre i raggruppamenti si ottengono con le parentesi graffe ('`{ }`'). I commenti ricalcano la forma usata per il linguaggio C:

```
/* commento_generico */
```

```
// commento_fino_alla_fine_della_riga
```

La gestione delle variabili di JavaScript è semplificata, al punto che non esiste la dichiarazione del tipo ed eventualmente la conversione avviene in modo automatico (salva la necessità di eseguire delle conversioni specifiche, quando quelle automatiche non avvengono nel modo desiderato).

Il punto critico della compatibilità dei programmi realizzati in JavaScript sta nell'utilizzo di oggetti (con le proprietà e i metodi relativi) che sono prestabiliti e creati automaticamente. Evidentemente, la disponibilità di questi oggetti dipende dal programma usato per interpretare il linguaggio JavaScript, ovvero dal navigatore; pertanto si può contare solo su un numero limitato di questi, così come bi-

sogna essere prudenti nell'uso delle proprietà e dei metodi relativi, anche se ci possono essere dei navigatori o altri interpreti in grado di accettare l'utilizzo di oggetti, istanze e metodi molto sofisticati.

Teoricamente è possibile creare degli oggetti nuovi, ma in pratica si tratta di un procedimento sconsigliabile, pertanto conviene limitarsi all'uso di quelli predefiniti, creando eventualmente delle funzioni personalizzate.

54.7.3 Variabili, costanti, tipi di dati ed espressioni

Le variabili possono essere dichiarate implicitamente, nel momento in cui vengono utilizzate, oppure si può usare una forma esplicita, ma in ogni caso non viene specificato il tipo del loro contenuto:

```
{var} nome{ = valore_assegnato};
```

Pertanto, la variabile ***x*** può essere dichiarata implicitamente così:

```
x = 1;
```

Oppure, si può usare la dichiarazione esplicita:

```
var x = 1;
```

I nomi delle variabili possono essere composti da lettere e numeri, che devono iniziare necessariamente con una lettera. Volendo seguire la convenzione del linguaggio Java, i nomi delle variabili si possono comporre unendo assieme una serie di parole che consentono di intendere il ruolo delle variabili stesse, utilizzando l'iniziale maiuscola per ogni parola che compone l'insieme del nome, tranne che per la prima, che inizia senza iniziale maiuscola. Per esempio: '**`miaStringa`**', '**`elencoNomiUtenti`**', '**`indiceElenco`**'.

I tipi di dati principali sono molto pochi, dal momento che nella gestione dei numeri si distingue soltanto tra interi e numeri con virgola mobile.

Tabella 54.116. Tipi di dati principali e rappresentazione delle costanti.

Tipo	Esempi di costante
intero	'**`0`**', '**`123`**', '**`45678`**'
virgola mobile	'**`0.1`**', '**`123.45`**', '**`45678.901`**'
booleano	'**`true`**', '**`false`**'
stringa	'**`"bla bla bla"`**', '**`"0422.1234567"`**', '**`'bene bene'`**'

Dagli esempi mostrati nella tabella riepilogativa dei tipi principali, si può osservare che i valori numerici in virgola mobile utilizzano il punto per separare la parte intera da quella decimale; inoltre, le stringhe sono delimitate indifferentemente con apici doppi oppure singoli. Nelle tabelle successive si riepilogano gli operatori principali delle espressioni che si possono realizzare con JavaScript.

Tabella 54.117. Elenco degli operatori aritmetici e di quelli di assegnamento relativi a valori numerici.

Operatore e operandi	Descrizione
`++`***op***	Incrementa di un'unità l'operando prima che venga restituito il suo valore.
op`++`	Incrementa di un'unità l'operando dopo averne restituito il suo valore.
`--`***op***	Decrementa di un'unità l'operando prima che venga restituito il suo valore.
op`--`	Decrementa di un'unità l'operando dopo averne restituito il suo valore.
`+`***op***	Non ha alcun effetto.
`-`***op***	Inverte il segno dell'operando.
op1 `+` ***op2***	Somma i due operandi.
op1 `-` ***op2***	Sottrae dal primo il secondo operando.
op1 `*` ***op2***	Moltiplica i due operandi.

Operatore e operandi	Descrizione
op1 `/` ***op2***	Divide il primo operando per il secondo.
op1 `%` ***op2***	Modulo -- il resto della divisione tra il primo e il secondo operando.
var `=` ***valore***	Assegna alla variabile il valore alla destra.
op1 `+=` ***op2***	***op1*** `=` ***op1*** `+` ***op2***
op1 `-=` ***op2***	***op1*** `=` ***op1*** `-` ***op2***
op1 `*=` ***op2***	***op1*** `=` ***op1*** `*` ***op2***
op1 `/=` ***op2***	***op1*** `=` ***op1*** `/` ***op2***
op1 `%=` ***op2***	***op1*** `=` ***op1*** `%` ***op2***

Tabella 54.118. Elenco degli operatori di confronto. Le metavariabili indicate rappresentano gli operandi e la loro posizione.

Operatore e operandi	Descrizione
op1 `==` ***op2***	*Vero* se gli operandi si equivalgono.
op1 `!=` ***op2***	*Vero* se gli operandi sono differenti.
op1 `<` ***op2***	*Vero* se il primo operando è minore del secondo.
op1 `>` ***op2***	*Vero* se il primo operando è maggiore del secondo.
op1 `<=` ***op2***	*Vero* se il primo operando è minore o uguale al secondo.
op1 `>=` ***op2***	*Vero* se il primo operando è maggiore o uguale al secondo.

Tabella 54.119. Elenco degli operatori logici. Le metavariabili indicate rappresentano gli operandi e la loro posizione.

Operatore e operandi	Descrizione
`!` ***op***	Inverte il risultato logico dell'operando.
op1 `&&` ***op2***	Se il risultato del primo operando è *Falso* non valuta il secondo.
op1 `||` ***op2***	Se il risultato del primo operando è *Vero* non valuta il secondo.

Tabella 54.120. Concatenamento di stringhe. Le metavariabili indicate rappresentano gli operandi e la loro posizione.

Operatore e operandi	Descrizione
stringa `+` ***op***	Crea una stringa unica concatenando la stringa a sinistra con il valore a destra, previa conversione di tipo, se necessaria.
op `+` ***stringa***	Crea una stringa unica concatenando il valore a sinistra, previa conversione di tipo, se necessaria, assieme alla stringa che si trova alla destra.

54.7.4 Funzioni e campo di azione delle variabili

Come già accennato nel capitolo, il linguaggio JavaScript offre oggetti già pronti, dei quali si usano le proprietà e i metodi relativi (ma con prudenza, per motivi di compatibilità), quindi consente di realizzare delle funzioni, che si dichiarano in modo molto semplice:

```
function nome ([parametro[, parametro]…]) {
    istruzione
    …
}
```

Come si può intendere, nel modello mostrato le parentesi graffe fanno parte della dichiarazione della funzione e servono a raggruppare le istruzioni che questa contiene.

Una funzione può restituire un valore; in tal caso si usa l'istruzione '**return**':

```
return valore;
```

Le variabili dichiarate all'interno della funzione hanno una campo di azione locale e oscurano temporaneamente variabili globali, con lo stesso nome, dichiarate al di fuori della funzione. Le variabili dichiarate al di fuori delle funzioni, ma prima della loro chiamata, sono accessibili all'interno delle stesse, se non sono oscurate.

54.7.5 Strutture di controllo di flusso

Le strutture di controllo di flusso disponibili nel linguaggio JavaScript sono sostanzialmente le stesse del linguaggio Java, tenendo conto però che alcuni navigatori potrebbero non riconoscere le strutture più sofisticate. Vengono mostrati i modelli sintattici relativi alle strutture più comuni e più compatibili, con qualche esempio e poche spiegazioni.

- ```
  if (condizione) istruzione
  ```

  ```
  if (condizione) istruzione else istruzione
  ```

  Se la condizione si verifica, viene eseguita l'istruzione (o il gruppo di istruzioni) seguente; quindi il controllo passa alle istruzioni successive alla struttura. Se viene utilizzato '**else**', nel caso non si verifichi la condizione, viene eseguita l'istruzione che ne segue. Vengono mostrati alcuni esempi:

  ```
  var importo;
  …
  if (importo > 10000000) document.write ("Offerta vantaggiosa");
  ```

  ```
  var importo;
  var memorizza;
  …
  if (importo > 10000000)
    {
      memorizza = importo;
      document.write ("Offerta vantaggiosa");
    }
  else
    {
      document.write ("Non conviene");
    }
  ```

  ```
  var importo;
  var memorizza;
  …
  if (importo > 10000000)
    {
      memorizza = importo;
      document.write ("Offerta vantaggiosa");
    }
  else if (importo > 5000000)
    {
      memorizza = importo;
      document.write ("Offerta accettabile");
    }
  else
    {
      document.write ("Non conviene");
    }
  ```

- ```
  while (condizione) istruzione
  ```

  Il comando '**while**' esegue un'istruzione, o un gruppo di queste, finché la condizione continua a restituire il valore *Vero*. La condizione viene valutata prima di eseguire il gruppo di istruzioni e poi ogni volta che termina un ciclo, prima dell'esecuzione del successivo. L'esempio seguente visualizza 10 volte la lettera «x»:


```
var contatore = 0;
while (contatore < 10)
  {
    contatore++;
    document.write ("x");
  }
```

Nel blocco di istruzioni di un ciclo '**while**', ne possono apparire alcune particolari:

– '**break**'
  esce definitivamente dal ciclo '**while**';
– '**continue**'
  interrompe l'esecuzione del gruppo di istruzioni e riprende dalla valutazione della condizione.

L'esempio seguente è una variante del ciclo di visualizzazione mostrato sopra, modificato in modo da vedere il funzionamento di '**break**'. Si osservi che '**while (true)**' equivale a un ciclo senza fine, perché la condizione è sempre vera:

```
var contatore = 0;

while (true)
  {
    if (contatore >= 10)
      {
        break;
      }
    contatore++;
    document.write ("x");
  }
```

- ```
  do blocco_di_istruzioni while (condizione);
  ```

  Il comando '**do**' esegue un gruppo di istruzioni una volta e poi ne ripete l'esecuzione finché la condizione continua a restituire il valore *Vero*.

- ```
  for (espressione1; espressione2; espressione3) istruzione
  ```

  Questa è la forma tipica di un'istruzione '**for**', in cui la prima espressione corrisponde all'assegnamento iniziale di una variabile, la seconda a una condizione che deve verificarsi fino a che si vuole che sia eseguita l'istruzione (o il gruppo di istruzioni), mentre la terza serve per l'incremento o decremento della variabile inizializzata con la prima espressione. In pratica, potrebbe esprimersi nella sintassi seguente:

  ```
  for (var = n; condizione; var++) istruzione
  ```

  Il ciclo '**for**' potrebbe essere definito anche in maniera differente, più generale: la prima espressione viene eseguita una volta sola all'inizio del ciclo; la seconda viene valutata all'inizio di ogni ciclo e il gruppo di istruzioni viene eseguito solo se il risultato è *Vero*; l'ultima viene eseguita alla fine dell'esecuzione del gruppo di istruzioni, prima che si ricominci con l'analisi della condizione.

  L'esempio in cui viene visualizzata per 10 volte una «x», potrebbe tradursi nel modo seguente, attraverso l'uso di un ciclo '**for**':

  ```
  var contatore;

  for (contatore = 0; contatore < 10; contatore++)
    {
      document.write ("x");
    }
  ```

54.7.6 Array

A differenza degli altri tipi di dati mostrati, le variabili che devono fare riferimento a un array devono essere dichiarate come tali:

```
var nome = new Array();
```

Si osservi che le variabili dichiarate in questo modo **fanno riferimento** a un array, pertanto non contengono direttamente l'array stesso. Ciò significa che, se il nome di un array viene indicato in una chiamata di funzione, se la funzione modifica il contenuto dell'array, la modifica riguarda lo stesso array a cui fa riferimento la variabile nella chiamata.

Gli elementi di un array dichiarato in questo modo vengono creati automaticamente nel momento in cui sono utilizzati, come si vede nell'esempio seguente:

```
var mioArray = new Array();
//
mioArray[0] = "Ciao";
mioArray[1] = "come";
mioArray[7] = "stai";
//
document.write (mioArray[0]);
document.write (" ");
document.write (mioArray[1]);
document.write (" ");
document.write (mioArray[7]);
```

In questo caso sono stati saltati volutamente gli elementi dall'indice due all'indice sei, che esistono, ma contenenti un valore indefinito.

Nel linguaggio JavaScript, così come in Java, gli array sono degli oggetti e come tali possono disporre di proprietà e di metodi. Quando gli elementi dell'array vengono dichiarati in questo modo, cioè con un indice numerico intero, è possibile leggere il metodo '**`length`**', che consente di conoscere la lunghezza dell'array stesso:

```
var mioArray = new Array();
//
mioArray[0] = "Ciao";
mioArray[1] = "come";
mioArray[7] = "stai";
//
document.write (mioArray.length);
```

In questo caso, il valore che viene visualizzato attraverso ***document.write()*** è il numero otto, dal momento che array contiene, formalmente, otto elementi.

L'indice dell'array può essere indicato attraverso una stringa, come nell'esempio seguente:

```
var mioArray = new Array();
//
mioArray["a"] = "Ciao";
mioArray["s"] = "come";
mioArray["d"] = "stai";
//
document.write (mioArray["a"]);
document.write (" ");
document.write (mioArray["s"]);
document.write (" ");
document.write (mioArray["d"]);
```

Questi indici sono o diventano dei metodi dell'array, consentendo una forma più semplice per indicare i vari elementi:

```
var mioArray = new Array();
//
mioArray["a"] = "Ciao";
mioArray.s    = "come";
mioArray.d    = "stai";
//
document.write (mioArray.a);
document.write (" ");
document.write (mioArray["s"]);
document.write (" ");
document.write (mioArray["d"]);
```

In pratica, indicare '**`mioArray["a"]`**', oppure '**`mioArray.a`**', dovrebbe essere la stessa cosa.

La proprietà '**`length`**' dovrebbe funzionare solo per gli elementi numerati in modo tradizionale, ignorando quelli a cui si accede per mezzo di un indice differente.

54.7.7 Funzioni standard

Alcune «funzioni» standard facilitano l'uso del linguaggio. Nelle tabelle successive si riepilogano quelle più comuni che dovrebbero essere compatibili nella maggior parte delle situazioni.

Tabella 54.131. Funzioni generali, ovvero metodi che non richiedono l'indicazione di un oggetto.

Funzione	Descrizione
`escape(`***stringa***`)`	Restituisce la stringa indicata come argomento, dopo averla trasformata in modo da poterla rappresentare in un indirizzo URI. La trasformazione implica l'uso delle sequenze formate con il prefisso '**`%`**'.
`unescape(`***stringa***`)`	Restituisce la stringa indicata come argomento, dopo averla trasformata in modo da tradurre le sequenze formate con il prefisso '**`%`**' nei caratteri che queste rappresentano. In pratica è l'inverso di ***escape()***.
`parseFloat(`***stringa***`)`	Elabora la stringa restituendo il numero a virgola mobile che questa rappresenta (o che dovrebbe rappresentare).
`parseInt(`***stringa***`)`	Elabora la stringa restituendo il numero intero che questa rappresenta (o che dovrebbe rappresentare).
`eval(`***stringa***`)`	Esegue il contenuto della stringa come un'istruzione e restituisce il valore che questa istruzione restituisce a sua volta.
`isNaN(`***valore***`)`	Verifica un valore e restituisce *Vero* se questo risulta essere indefinito.

Tabella 54.132. Metodi matematici, che si applicano necessariamente all'oggetto '**`Math`**', come mostrato in modo esplicito nel modello sintattico.

Metodo	Descrizione
`Math.abs(`***n***`)`	Restituisce il valore assoluto di ***n***
`Math.ceil(`***n***`)`	Restituisce il valore intero di ***n*** arrotondato per eccesso.
`Math.round(`***n***`)`	Restituisce il valore intero di ***n*** arrotondato per eccesso se ***n*** $\geq 0{,}5$, altrimenti arrotondato per difetto.
`Math.floor(`***n***`)`	Restituisce il valore intero di ***n*** arrotondato per difetto.
`Math.exp(`***n***`)`	Restituisce e^{n} (dove «e» approssima il valore 2,718).
`Math.log(`***n***`)`	Restituisce il logaritmo naturale di ***n***.
`Math.max(`***n_1, n_2, …n_n***`)`	Restituisce il massimo tra i valori numerici indicati.
`Math.min(`***n_1, n_2, …n_n***`)`	Restituisce il minimo tra i valori numerici indicati.
`Math.pow(`***x, y***`)`	Restituisce il risultato di x^{y}.
`Math.random()`	Restituisce un numero casuale, a virgola mobile, nell'intervallo compreso tra zero e uno.
`Math.sqrt(`***n***`)`	Restituisce la radice quadrata di ***n***.
`Math.sin(`***n***`)`	Restituisce il seno di ***n***.
`Math.cos(`***n***`)`	Restituisce il coseno di ***n***.
`Math.tan(`***n***`)`	Restituisce la tangente di ***n***.
`Math.asin(`***n***`)`	Restituisce l'arcoseno di ***n***.
`Math.acos(`***n***`)`	Restituisce l'arcocoseno di ***n***.
`Math.atan(`***n***`)`	Restituisce l'arcotangente di ***n***.

Tabella 54.133. Istruzioni per la creazione di oggetti contenenti informazioni data-orario.

Istruzione	Descrizione
`x = new Date()`	Costruisce l'oggetto ***x*** contenente la data e l'orario corrente.
`x = new Date(anno, mese, giorno)` `x = new Date(anno, mese, giorno, ore, minuti, secondi)`	Costruisce l'oggetto ***x*** contenente la data ed eventualmente anche l'orario indicati.

Tabella 54.134. Metodi per leggere o modificare informazioni data-orario a oggetti creati con ***Date()***.

Metodo	Descrizione
`x.setYear(anno)`	Imposta l'anno nell'oggetto ***x***.
`x.setMonth(anno)`	Imposta il mese nell'oggetto ***x***.
`x.setDate(giorno)`	Imposta il giorno del mese nell'oggetto ***x***.
`x.setHours(ora)`	Imposta l'ora nell'oggetto ***x***.
`x.setMinutes(minuti)`	Imposta i minuti nell'oggetto ***x***.
`x.setSeconds(secondi)`	Imposta i secondi nell'oggetto ***x***.
`x.getYear()`	Restituisce l'anno contenuto nell'oggetto ***x***.
`x.getMonth()`	Restituisce il mese contenuto nell'oggetto ***x***.
`x.getDay()`	Restituisce il giorno contenuto nell'oggetto ***x***.
`x.getHours()`	Restituisce l'ora contenuta nell'oggetto ***x***.
`x.getMinutes()`	Restituisce i minuti contenuti nell'oggetto ***x***.
`x.getSeconds()`	Restituisce i secondi contenuti nell'oggetto ***x***.

Si osservi che l'anno potrebbe essere restituito in forma non corretta; per esempio, l'anno 2004 potrebbe risultare come il numero 104; nello stesso modo, l'anno 1904 potrebbe apparire come il numero 4.

Tabella 54.135. Funzioni varie.

Funzione	Descrizione
`alert(stringa)`	Mostra la stringa all'interno di una finestra di avvertimento.
`setTimeout(stringa, n)`	Esegue la funzione scritta nella stringa che rappresenta il primo parametro, dopo aver atteso ***n*** ms (millesimi di secondo).

54.7.8 Gestione delle stringhe

Anche per JavaScript, come per Java, le stringhe sono oggetti, per i quali esistono metodi appropriati ed esiste anche la proprietà '**`length`**', che restituisce la lunghezza della stringa (come già avviene per gli array). Molti dei metodi riferiti alle stringhe servono per delimitare il testo con dei marcatori appropriati, per esempio per ottenere un testo nero o un corsivo, o ancora qualcosa di più appariscente.

Tabella 54.136. Metodi relativi alla gestione delle stringhe.

Metodo	Descrizione
`x.charAt(n)`	Restituisce un carattere nella posizione ***n*** della stringa contenuta nell'oggetto ***x***, dove il primo carattere viene identificato con lo zero.
`x.substring(n, m)`	Restituisce la sottostringa ottenuta a partire alla posizione ***n*** fino alla posizione ***m*** esclusa.

Metodo	Descrizione
`x.indexOf(stringa[, n])`	Restituisce la posizione a partire dalla quale la stringa indicata come primo parametro corrisponde con la stringa contenuta nell'oggetto ***x***. Se è indicato anche il secondo parametro, la ricerca parte dalla posizione ***n***. Se la corrispondenza non viene trovata, restituisce il valore `-1`.
`x.lastIndexOf(stringa[, n])`	Restituisce la posizione più a destra dove la stringa indicata come primo parametro corrisponde con la stringa contenuta nell'oggetto ***x***. Se è indicato anche il secondo parametro, la ricerca termina in corrispondenza dalla posizione ***n***. Se la corrispondenza non viene trovata, restituisce il valore `-1`.
`x.toLowerCase()`	Restituisce la stringa contenuta nell'oggetto ***x*** convertendola in lettere minuscole.
`x.toUpperCase()`	Restituisce la stringa contenuta nell'oggetto ***x*** convertendola in lettere maiuscole.
`x.fontcolor(colore)`	Restituisce la stringa contenuta nell'oggetto ***x***, aggiungendo i marcatori '**`<font color="`*colore*`">`**' e '**`</font>`**' all'inizio e alla fine della stessa.
`x.fontsize(dimensione)`	Restituisce la stringa contenuta nell'oggetto ***x***, aggiungendo i marcatori '**`<font size="`*dimensione*`">`**' e '**`</font>`**' all'inizio e alla fine della stessa.
`x.blink()`	Restituisce la stringa contenuta nell'oggetto ***x***, aggiungendo i marcatori '**`<blink>`**' e '**`</blink>`**' all'inizio e alla fine della stessa.
`x.bold()`	Restituisce la stringa contenuta nell'oggetto ***x***, aggiungendo i marcatori '**`<b>`**' e '**`</b>`**' all'inizio e alla fine della stessa.
`x.italics()`	Restituisce la stringa contenuta nell'oggetto ***x***, aggiungendo i marcatori '**`<i>`**' e '**`</i>`**' all'inizio e alla fine della stessa.
`x.fixed()`	Restituisce la stringa contenuta nell'oggetto ***x***, aggiungendo i marcatori '**`<tt>`**' e '**`</tt>`**' all'inizio e alla fine della stessa.
`x.big()`	Restituisce la stringa contenuta nell'oggetto ***x***, aggiungendo i marcatori '**`<big>`**' e '**`</big>`**' all'inizio e alla fine della stessa.
`x.small()`	Restituisce la stringa contenuta nell'oggetto ***x***, aggiungendo i marcatori '**`<small>`**' e '**`</small>`**' all'inizio e alla fine della stessa.
`x.sub()`	Restituisce la stringa contenuta nell'oggetto ***x***, aggiungendo i marcatori '**`<sub>`**' e '**`</sub>`**' all'inizio e alla fine della stessa.
`x.sup()`	Restituisce la stringa contenuta nell'oggetto ***x***, aggiungendo i marcatori '**`<sup>`**' e '**`</sup>`**' all'inizio e alla fine della stessa.
`x.strike()`	Restituisce la stringa contenuta nell'oggetto ***x***, aggiungendo i marcatori '**`<strike>`**' e '**`</strike>`**' all'inizio e alla fine della stessa.
`x.anchor(nome)`	Restituisce la stringa contenuta nell'oggetto ***x***, aggiungendo i marcatori '**`<a name="`*nome*`">`**' e '**`</a>`**' all'inizio e alla fine della stessa.

Metodo	Descrizione
`x.link(`*riferimento*`)`	Restituisce la stringa contenuta nell'oggetto ***x***, aggiungendo i marcatori '**`<a href="`*riferimento*`">`**' e '**`</a>`**' all'inizio e alla fine della stessa.

54.7.9 Moduli «FORM»

I dati che vengono inseriti all'interno di moduli HTML (elementi '**`FORM`**'), possono essere letti come proprietà di oggetti che discendono da '**`document`**'. All'interno di un elemento '**`FORM`**', si inseriscono normalmente una serie di elementi '**`INPUT`**', con caratteristiche differenti; sia l'elemento '**`FORM`**', sia l'elemento '**`INPUT`**' hanno la possibilità di avere un nome, attraverso l'attributo '**`NAME`**'. Nella situazione più semplice, si può avere un elemento '**`INPUT`**' per l'inserimento manuale di un testo da parte dell'utente, come nell'esempio seguente, dove per il momento l'attributo '**`NOME`**' dell'elemento '**`FORM`**' non sarebbe indispensabile:

```
<FORM NAME="elaborazione" METHOD="get"
 ACTION="/cgi-bin/elabora">
<P><INPUT NAME="nominativo" TYPE="TEXT" SIZE="30">
<INPUT TYPE="submit" VALUE="Invia"></P>
</FORM>
```

Un programma JavaScript ha la possibilità di leggere il testo inserito nel campo '**`nominativo`**' attraverso la proprietà '**`document.elaborazione.nominativo.value`**'; in pratica, sono stati usati i valori degli attributi '**`NAME`**' per costruire l'oggetto relativo. Per esempio, si potrebbe modificare l'esempio nel modo seguente:

```
<FORM METHOD="get" ACTION="/cgi-bin/elabora">
<P><INPUT TYPE="TEXT" SIZE="30" NAME="nominativo">
<INPUT TYPE="button" VALUE="Invia"
 onClick="verificaInvia()"></P>
</FORM>
```

Come si vede, il bottone finale è stato modificato, da un tipo '**`submit`**' a un tipo '**`button`**', per evitare che l'informazione venga inviata con la sola selezione del bottone, che invece provoca l'esecuzione della funzione ***verificaInvia()***, che intuitivamente ha lo scopo di verificare i dati inseriti e di inviarli se corretti. La funzione di controllo potrebbe essere realizzata così:

```
function verificaInvia () {
    if (document.elaborazione.nominativo.value == "")
      {
        alert ("È necessario inserire il nominativo!");
        document.elaborazione.nominativo.focus();
      }
    else
      {
        document.elaborazione.submit();
      }
}
```

Intuitivamente, osservando l'esempio, si comprende che se il campo risulta vuoto nel momento in cui si seleziona il bottone, viene mostrato un avvertimento e viene messo a fuoco proprio il campo da correggere, altrimenti viene inviato il contenuto del modulo '**`elaborazione`**'.

In considerazione del fatto che non si può avere la certezza di avere un utenza che può disporre in ogni circostanza di un navigatore compatibile con il programma JavaScript che si scrive, si può essere più morbidi e dare la possibilità di inviare i dati senza alcun controllo:

```
<FORM METHOD="get" ACTION="/cgi-bin/elabora">
<P><INPUT TYPE="TEXT" SIZE="30" NAME="nominativo">
<INPUT TYPE="button" VALUE="Invia"
 onClick="verificaInvia()"></P>
<INPUT TYPE="submit" VALUE="Invia senza controllo"></P>
</FORM>
```

Nelle tabelle successive vengono descritti alcuni eventi, metodi e proprietà relative all'uso di elementi '**`FORM`**'.

Tabella 54.141. Eventi relativi all'elemento '**`FORM`**' e all'elemento '**`INPUT`**'.

Modello	Descrizione
`<FORM … onSubmit="`*funzione*`">`	Esegue la funzione quando il modulo viene inviato.
`<FORM … onReset="`*funzione*`">`	Esegue la funzione quando il modulo viene azzerato.
`<INPUT … onFocus="`*funzione*`">`	Esegue la funzione quando il componente è a fuoco.
`<INPUT … onBlur="`*funzione*`">`	Esegue la funzione quando il componente non è più a fuoco.
`<INPUT … onClick="`*funzione*`">`	Esegue la funzione quando il componente viene selezionato con un clic.
`<INPUT … onSelect="`*funzione*`">`	Esegue la funzione quando viene selezionata una porzione del testo del componente.

Tabella 54.142. Alcune proprietà e metodi riferiti a oggetti derivanti da elementi '**`FORM`**' e '**`INPUT`**'. La metavariabile ***x*** rappresenta il nome associato all'elemento '**`FORM`**' e ***y*** rappresenta il nome associato all'elemento '**`INPUT`**'.

Modello	Descrizione
`document.`***x***`.action`	Rappresenta l'indirizzo usato normalmente nell'attributo '**`ACTION`**' dell'elemento '**`FORM`**'.
`document.`***x***`.encoding`	Codifica usata nel modulo.
`document.`***x***`.method`	Modalità di invio dei dati del modulo ('**`GET`**' o '**`POST`**').
`document.`***x***`.submit()`	Invia i dati del modulo.
`document.`***x***`.reset()`	Azzera il contenuto del modulo.
`document.`***x***`.`***y***`.type`	Restituisce il tipo di campo.
`document.`***x***`.`***y***`.value`	Restituisce il valore del campo.
`document.`***x***`.`***y***`.focus()`	Seleziona l'elemento (lo mette a fuoco).
`document.`***x***`.`***y***`.blur()`	Deseleziona l'elemento (l'opposto della messa a fuoco).

Dovendo lavorare con nomi di proprietà e metodi molto lunghi, può essere conveniente dichiarare il contesto con l'istruzione '**`with`**':

```
with (prefisso) {
    …
}
```

L'esempio già apparso della funzione ***verificaInvia()*** potrebbe essere semplificato agli occhi del lettore modificandolo così:

```
function verificaInvia () {
    with (document.elaborazione)
      {
        if (nominativo.value == "")
          {
            alert ("È necessario inserire il nominativo!");
            nominativo.focus();
          }
        else
          {
            submit();
          }
      }
}
```

54.7.10 Esempi di programmazione

Si tenga in considerazione il fatto che in condizioni normali non si ha la disponibilità di un analizzatore sintattico del linguaggio JavaScript, pertanto la ricerca di errori banali di sintassi diventa problematica, dal momento che il navigatore comune si limita a non eseguire il programma che gli risulta non essere corretto. Anche per questa ragione non vengono mostrati esempi di una certa complessità.

54.7.10.1 Fattoriale

```
<!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
<HTML LANG="it">
<HEAD>
    <META HTTP-EQUIV="Content-Type" CONTENT="text/html">
    <TITLE>Calcola il fattoriale di un valore intero</TITLE>
    <SCRIPT LANGUAGE="JavaScript" TYPE="text/javascript">
    <!--
        function fattoriale (x)
        {
            var i = x - 1;
            //
            while (i > 0)
              {
                x = x * i;
                i--;
              }
            return x;
        }
        //
        function esegui ()
        {
            var x = parseInt (document.modulo.x.value);
            var z;
            var risultato = "";
            z = fattoriale (x);
            //
            risultato = risultato
                      + x
                      + "!"
                      + "="
                      + z;
            alert (risultato);
        }
        //-->
        </SCRIPT>
</HEAD>
<BODY>
<H1>Calcola il fattoriale di un valore intero</H1>

<FORM NAME="modulo" METHOD="get" ACTION="">
<P>
    x = <INPUT TYPE="TEXT" SIZE="30" NAME="x">
</P>
<P>
    <INPUT TYPE="button" VALUE="esegui" onClick="esegui()">
</P>
</FORM>

</BODY>
</HTML>
```

In alternativa, l'algoritmo si può tradurre in modo ricorsivo:

```
function fattoriale (x)
{
    if (x > 1)
      {
        return (x * fattoriale (x - 1));
      }
    else
      {
        return 1;
      }
    //
    // Teoricamente non dovrebbe arrivare qui.
    //
}
```

54.7.10.2 Massimo comune divisore

```
<!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
<HTML LANG="it">
<HEAD>
    <META HTTP-EQUIV="Content-Type" CONTENT="text/html">

    <TITLE>Determina il massimo comune divisore tra due
           numeri interi positivi</TITLE>

    <SCRIPT LANGUAGE="JavaScript" TYPE="text/javascript">
    <!--
        function mcd (x, y)
        {
            var i;
            var z = 0;
            //
            while (x != y)
              {
                if (x > y)
                  {
                    x = x - y;
                  }
                else
                  {
                    y = y - x;
                  }
              }
            return x;
        }
        //
        function esegui ()
        {
            var x = parseInt (document.modulo.z.value);
            var y = parseInt (document.modulo.y.value);
            var z;
            var risultato = "";
            z = mcd (x, y);
            //
            risultato = risultato
                      + "mcd("
                      + x
                      + ", "
                      + y
                      + ")"
                      + "="
                      + z;
            alert (risultato);
        }
        //-->
        </SCRIPT>
</HEAD>
<BODY>
<H1>Determina il massimo comune divisore tra due numeri
interi positivi</H1>

<FORM NAME="modulo" METHOD="get" ACTION="">
<P>
    x = <INPUT TYPE="TEXT" SIZE="30" NAME="x"><BR>
    y = <INPUT TYPE="TEXT" SIZE="30" NAME="y">
</P>
<P>
    <INPUT TYPE="button" VALUE="esegui" onClick="esegui()">
</P>
</FORM>

</BODY>
</HTML>
```

54.7.10.3 Numero primo

```
<!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
<HTML LANG="it">
<HEAD>
    <META HTTP-EQUIV="Content-Type" CONTENT="text/html">
    <TITLE>Determina se un numero intero sia primo o
    meno</TITLE>
    <SCRIPT LANGUAGE="JavaScript" TYPE="text/javascript">
    <!--
        function primo (x)
        {
            var primo = true;
            var i = 2;
            var j;
            //
            while ((i < x) && primo)
            {
                j = x / i;
                j = Math.floor (j);
                j = x - (j * i);
                //
                if (j == 0)
                  {
                    primo = false;
```

```
                }
              else
                {
                  i++;
                }
          }
          return primo;
      }
      //
      function esegui ()
      {
          var x = parseInt (document.modulo.x.value);
          var z;
          var risultato = "";
          z = primo (x);
          //
          if (z)
            {
              risultato = risultato
                    + x
                    + " è un numero primo!";
            }
          else
            {
              risultato = risultato
                    + x
                    + " non è un numero primo!";
            }
          alert (risultato);
      }
      //-->
      </SCRIPT>
</HEAD>
<BODY>
<H1>Determina se un numero intero sia primo o meno</H1>

<FORM NAME="modulo" METHOD="get" ACTION="">
<P>
    x = <INPUT TYPE="TEXT" SIZE="30" NAME="x">
</P>
<P>
    <INPUT TYPE="button" VALUE="esegui" onClick="esegui()">
</P>
</FORM>

</BODY>
</HTML>
```

54.7.10.4 Ricerca sequenziale

```
<!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
<HTML LANG="it">
<HEAD>
    <META HTTP-EQUIV="Content-Type" CONTENT="text/html">
    <TITLE>Ricerca sequenziale</TITLE>
    <SCRIPT LANGUAGE="JavaScript" TYPE="text/javascript">
    <!--
        function ricercaseq (lista, x, a, z)
        {
            var i;
            //
            // Scandisce l'array alla ricerca dell'elemento.
            //
            for (i = a; i <= z; i++)
              {
                if (x == lista[i])
                  {
                    return i;
                  }
              }
            //
            // La corrispondenza non è stata trovata.
            //
            return -1;
        }
        function esegui ()
        {
            var a = new Array ();
            var z;
            var risultato = "";
            var x = document.modulo.x.value;
            //
            a[0] = document.modulo.a0.value;
            a[1] = document.modulo.a1.value;
            a[2] = document.modulo.a2.value;
            a[3] = document.modulo.a3.value;
            a[4] = document.modulo.a4.value;
            a[5] = document.modulo.a5.value;
            a[6] = document.modulo.a6.value;
            a[7] = document.modulo.a7.value;
            a[8] = document.modulo.a8.value;
            a[9] = document.modulo.a9.value;
            //
            z = ricercaseq (a, x, 0, (a.length-1));
            //
            risultato = x
                      + " si trova nella posizione "
                      + z
                      + ".";
            alert (risultato);
        }
        //-->
        </SCRIPT>
</HEAD>
<BODY>
<H1>Ricerca sequenziale</H1>

<FORM NAME="modulo" METHOD="get" ACTION="">
<P>
    x = <INPUT TYPE="TEXT" SIZE="30" NAME="x"><BR>
    a[0] = <INPUT TYPE="TEXT" SIZE="30" NAME="a0"><BR>
    a[1] = <INPUT TYPE="TEXT" SIZE="30" NAME="a1"><BR>
    a[2] = <INPUT TYPE="TEXT" SIZE="30" NAME="a2"><BR>
    a[3] = <INPUT TYPE="TEXT" SIZE="30" NAME="a3"><BR>
    a[4] = <INPUT TYPE="TEXT" SIZE="30" NAME="a4"><BR>
    a[5] = <INPUT TYPE="TEXT" SIZE="30" NAME="a5"><BR>
    a[6] = <INPUT TYPE="TEXT" SIZE="30" NAME="a6"><BR>
    a[7] = <INPUT TYPE="TEXT" SIZE="30" NAME="a7"><BR>
    a[8] = <INPUT TYPE="TEXT" SIZE="30" NAME="a8"><BR>
    a[9] = <INPUT TYPE="TEXT" SIZE="30" NAME="a9"><BR>
</P>
<P>
    <INPUT TYPE="button" VALUE="esegui" onClick="esegui()">
</P>
</FORM>

</BODY>
</HTML>
```

Esiste anche una soluzione ricorsiva che viene mostrata nella subroutine seguente:

```
function ricercaseq (lista, x, a, z)
{
    if (a > z)
      {
        //
        // La corrispondenza non è stata trovata.
        //
        return -1;
      }
    else if (x == lista[a])
      {
        return a;
      }
    else
      {
        return ricercaseq (lista, x, a+1, z);
      }
}
```

54.8 Approfondimento: verifiche automatiche con JavaScript

Questa sezione contiene la descrizione di un programma molto semplice, scritto con il linguaggio JavaScript, per la realizzazione di verifiche che generano automaticamente la valutazione. Il programma riguarda una verifica di esempio e serve solo come idea di un modello da utilizzare per la realizzazione di altre verifiche del genere. Si osservi che un meccanismo molto simile a quello descritto qui si

può utilizzare all'interno di Alml, il quale genera automaticamente il codice JavaScript necessario.

Viene mostrato subito il listato completo della pagina HTML contenente il programma; nelle sezioni successive viene descritto nelle sue varie componenti.

```
<HTML LANG="it">
<HEAD>
    <META HTTP-EQUIV="Content-Type" CONTENT="text/html">
    <TITLE>Verifica banale</TITLE>
    <SCRIPT LANGUAGE="JavaScript" TYPE="text/javascript">
    <!--
        //
        // Something to do at the beginning.
        //
        // Get the initial time stamp.
        //
        var start_time = new Date ();
        //
        // Countdown function.
        //
        function countdown (time_allowed)
        {
            var hours;
            var minutes;
            var seconds;
            var time;
            var time_string;
            var time_sign;
            //
            // Verify the countdown time sign.
            //
            if (time_allowed > 0)
              {
                time_sign = "";
              }
            else
              {
                time_sign = "-";
              }
            //
            // Find the absolute countdown value.
            //
            time = Math.abs (time_allowed);
            //
            hours   = (time - (time % 3600000)) / 3600000;
            time    = time - (hours * 3600000);
            minutes = (time - (time % 60000)) / 60000;
            time    = time - (minutes * 60000);
            seconds = (time - (time % 1000)) / 1000;
            //
            if (minutes < 10)
              {
                minutes = "0" + minutes;
              }
            if (seconds < 10)
              {
                seconds = "0" + seconds;
              }
            if (hours < 10)
              {
                hours = "0" + hours;
              }
            //
            time_string = time_sign + hours + ":" + minutes
                          + ":" + seconds;
            //
            // Show the time string inside some text field.
            //
            document.test_form_1.countdown_0.value
             = time_string;
            document.test_form_1.countdown_1.value
             = time_string;
            document.test_form_1.countdown_2.value
             = time_string;
            //
            // Reduce 10 s inside the countdown variable.
            //
            time_allowed = time_allowed - 10000;
            //
            // Wait for 10 s for the recursive call.
            //
            setTimeout ("countdown (" + time_allowed + ")",
                        10000);
            //
        }
        //
        // Radio button scan function.
        //
        function scan_radio_button (radio_button)
        {
            var counter;
            //
            for (counter = 0;
                 counter < radio_button.length;
                 counter++)
              {
                if (radio_button[counter].checked)
                  {
                    return (counter + 1);
                  }
              }
            //
            // Nothing found.
            //
            return 0;
        }
        //
        // Check test_form_1.
        //
        function check_test_form_1 (start_time)
        {
            //
            var end_time = new Date ();
            var test_time = end_time - start_time;
            var time_allowed = 1 * 60000;
            var time_delay = test_time - time_allowed;
            //
            if (time_delay < 0)
              {
                time_delay = 0;
              }
            //
            // Penalty for a minute delay.
            //
            var score_time_penalty = 0.5;
            //
            // Set the result for every choice.
            //
            var score_dita_di_una_mano = new Array();
            score_dita_di_una_mano[0] = 0;
            score_dita_di_una_mano[1] = 0;
            score_dita_di_una_mano[2] = 0;
            score_dita_di_una_mano[3] = 0;
            score_dita_di_una_mano[4] = 0;
            score_dita_di_una_mano[5] = 0;
            score_dita_di_una_mano[6] = 5;
            score_dita_di_una_mano[7] = 0;
            score_dita_di_una_mano[8] = 0;
            //
            // Set the result for every choice.
            //
            var score_dita_di_un_piede = new Array();
            score_dita_di_un_piede[0] = 0;
            score_dita_di_un_piede[1] = 0;
            score_dita_di_un_piede[2] = 0;
            score_dita_di_un_piede[3] = 0;
            score_dita_di_un_piede[4] = 0;
            score_dita_di_un_piede[5] = 0;
            score_dita_di_un_piede[6] = 5;
            score_dita_di_un_piede[7] = 0;
            score_dita_di_un_piede[8] = 0;
            //
            // Save the personal data.
            //
            var student
                = document.test_form_1.student.value;
            var date = document.test_form_1.date.value;
            //
            // Scan the radio buttons.
            //
            var dita_di_una_mano
```

```
            = scan_radio_button
                (document.test_form_1.dita_di_una_mano);
        //
        var dita_di_un_piede
            = scan_radio_button
                (document.test_form_1.dita_di_un_piede);
        //
        // Calculate the results.
        //
        var account_score_time_penalty;
        var account_score_dita_di_una_mano;
        var account_score_dita_di_un_piede;
        var account_score_sum = 0;
        //
        account_score_time_penalty
            = (time_delay / 1000)
               * (score_time_penalty / 60);
        //
        account_score_dita_di_una_mano
         = score_dita_di_una_mano[dita_di_una_mano];
        account_score_dita_di_un_piede
         = score_dita_di_un_piede[dita_di_un_piede];
        account_score_sum
         = account_score_sum
            - account_score_time_penalty;
        account_score_sum
         = account_score_sum
            + account_score_dita_di_una_mano;
        account_score_sum
         = account_score_sum
            + account_score_dita_di_un_piede;
        //
        // Reset the form to avoid the use of the [Back]
        // button from the browser.
        //
        document.test_form_1.reset();
        //
        // Print a simple HTML header.
        //
        document.writeln ("<HTML>");
        document.writeln ("<HEAD>");
        document.write   ("<TITLE>");
        document.write   ("Risultato della verifica");
        document.writeln ("</TITLE>");
        document.writeln ("</HEAD>");
        document.writeln ("<BODY>");
        //
        // Print a H1 title and some personal data.
        //
        document.writeln ("<H1>Verifica banale</H1>");
        document.writeln ("<P>studente: " + student
                          + "</P>");
        document.writeln ("<P>data: " + date + "</P>");
        //
        // Print about time.
        //
        document.write ("<P>inizio della verifica: ");
        document.write (start_time.getYear());
        document.write (".");
        document.write (start_time.getMonth() + 1);
        document.write (".");
        document.write (start_time.getDate());
        document.write (" ");
        document.write (start_time.getHours());
        document.write (":");
        document.write (start_time.getMinutes());
        document.write (".");
        document.write (start_time.getSeconds());
        document.writeln ("</P>");
        //
        document.write ("<P>conclusione della ");
        document.write ("verifica: ");
        document.write (end_time.getYear());
        document.write (".");
        document.write (end_time.getMonth() + 1);
        document.write (".");
        document.write (end_time.getDate());
        document.write (" ");
        document.write (end_time.getHours());
        document.write (":");
        document.write (end_time.getMinutes());
```



```
        document.write (".");
        document.write (end_time.getSeconds());
        document.writeln ("</P>");
        //
        document.write ("<P>durata complessiva della ");
        document.write ("verifica: ");
        document.write (test_time / 1000);
        document.write (" s, ");
        document.write (test_time / 1000 / 60);
        document.write (" m;");
        document.writeln ("</P>");
        //
        document.write ("<P>tempo a disposizione: ");
        document.write (time_allowed / 1000);
        document.write (" s; ");
        document.write ("ritardo: ");
        document.write (time_delay / 1000);
        document.write (" s; ");
        document.write ("penalit&agrave;/minuto: ");
        document.write (score_time_penalty);
        document.write ("; ");
        document.write ("penalit&agrave; ");
        document.write ("complessiva: ");
        document.write (account_score_time_penalty);
        document.writeln ("</P>");
        //
        // First question.
        //
        document.write ("<P>Quante sono le dita ");
        document.write ("di una mano? risposta n. ");
        document.write (dita_di_una_mano);
        document.write ("; ");
        document.write ("punteggio: ");
        document.write (account_score_dita_di_una_mano);
        document.writeln ("</P>");
        //
        // Second question.
        //
        document.write ("<P>Quante sono le dita ");
        document.write ("di un piede? risposta n. ");
        document.write (dita_di_un_piede);
        document.write ("; ");
        document.write ("punteggio: ");
        document.write (account_score_dita_di_un_piede);
        document.writeln ("</P>");
        //
        // Final result.
        //
        document.write ("<P>Punteggio complessivo ");
        document.write ("della verifica: ");
        document.write (account_score_sum);
        document.writeln ("</P>");
        //
        // End of HTML.
        //
        document.writeln ("</BODY>");
        document.writeln ("</HTML>");
        //
        // Print with a printer.
        //
        window.print ();
        //
        // Stop loading.
        //
        window.stop ();
    }
    //-->
    </SCRIPT>
</HEAD>
<BODY onLoad="document.test_form_1.reset();
              countdown (1 * 60000);">

<H1>Verifica banale</H1>

<FORM NAME="test_form_1" METHOD="get" ACTION="">

    <P>Tempo a disposizione: <INPUT TYPE="TEXT"
    NAME="countdown_0" SIZE="10" VALUE=" "></P>

    <P>Cognome, nome, classe:
    <INPUT TYPE="TEXT" SIZE="60" NAME="student" VALUE="">
```

```
    </P>

    <P>Data:
    <INPUT TYPE="TEXT" SIZE="60" NAME="date" VALUE="">
    </P>

    <P>Tempo a disposizione: <INPUT TYPE="TEXT"
    NAME="countdown_1" SIZE="10" VALUE=" "></P>

    <P><STRONG>Quante sono le dita di una mano?</STRONG></P>
    <OL>
    <LI><INPUT TYPE="RADIO"
               NAME="dita_di_una_mano">0 dita</LI>
    <LI><INPUT TYPE="RADIO"
               NAME="dita_di_una_mano">1 dita</LI>
    <LI><INPUT TYPE="RADIO"
               NAME="dita_di_una_mano">2 dita</LI>
    <LI><INPUT TYPE="RADIO"
               NAME="dita_di_una_mano">3 dita</LI>
    <LI><INPUT TYPE="RADIO"
               NAME="dita_di_una_mano">4 dita</LI>
    <LI><INPUT TYPE="RADIO"
               NAME="dita_di_una_mano">5 dita</LI>
    <LI><INPUT TYPE="RADIO"
               NAME="dita_di_una_mano">6 dita</LI>
    <LI><INPUT TYPE="RADIO"
               NAME="dita_di_una_mano">7 dita</LI>
    </OL>

    <P>Tempo a disposizione: <INPUT TYPE="TEXT"
    NAME="countdown_2" SIZE="10" VALUE=" "></P>

    <P><STRONG>Quante sono le dita di un piede?</STRONG></P>
    <OL>
    <LI><INPUT TYPE="RADIO"
               NAME="dita_di_un_piede">0 dita</LI>
    <LI><INPUT TYPE="RADIO"
               NAME="dita_di_un_piede">1 dita</LI>
    <LI><INPUT TYPE="RADIO"
               NAME="dita_di_un_piede">2 dita</LI>
    <LI><INPUT TYPE="RADIO"
               NAME="dita_di_un_piede">3 dita</LI>
    <LI><INPUT TYPE="RADIO"
               NAME="dita_di_un_piede">4 dita</LI>
    <LI><INPUT TYPE="RADIO"
               NAME="dita_di_un_piede">5 dita</LI>
    <LI><INPUT TYPE="RADIO"
               NAME="dita_di_un_piede">6 dita</LI>
    <LI><INPUT TYPE="RADIO"
               NAME="dita_di_un_piede">7 dita</LI>
    </OL>

    <P><INPUT TYPE="button" VALUE="concludi la verifica"
        onClick="check_test_form_1(start_time)"></P>

</FORM>

</BODY>
</HTML>
```

54.8.1 Utilizzo del programma

Il file mostrato è una pagina HTML che incorpora il codice JavaScript; per visualizzarla occorre un navigatore comune, in grado di eseguire il codice JavaScript. Potrebbe apparire come nella figura successiva:

Come si vede, appare un modello da compilare; al termine si deve selezionare il bottone `CONCLUDI LA VERIFICA` per ottenere l'esito. Ecco cosa si potrebbe ottenere se sono state date tutte le risposte in modo esatto, ma con un leggero ritardo rispetto ai tempi concessi:

In condizioni normali, dovrebbe apparire anche una finestra di conferma per la stampa dell'esito della verifica.

Si può osservare che la pagina normale della verifica viene azzerata completamente a ogni tentativo di ripristino della pagina e di azzeramento del tempo, in modo da scoraggiare un uso scorretto della verifica.

54.8.2 Codice HTML

Il codice HTML significativo è quello contenuto nell'elemento '**`BODY`**'. La prima cosa che si può osservare è l'associazione dell'evento '**`onLoad`**' alla chiamata della funzione ***document.test_form_1.reset()*** e successivamente della funzione ***countdown()***:

```
<BODY
  onLoad="document.test_form_1.reset(); countdown (1 * 60000);">
```

In pratica, la prima delle due funzioni esiste in quanto è stato definito il modulo (elemento '**`FORM`**') denominato '**`test_form_1`**' e serve ad azzerarne il contenuto; la seconda funzione è dichiarata nel codice JavaScript che appare nell'intestazione della pagina e il suo scopo è quello di azzerare il conto alla rovescia, in modo che parta da 60 000 ms (pari a 60 s, ovvero un minuto).

Con l'evento '**`onLoad`**' associato in questo modo alle due funzioni appena descritte, si vuole fare in modo che a ogni caricamento della pagina il suo contenuto si azzeri e il conteggio del tempo a disposizione riparta. Questa accortezza serve a evitare che si possa barare sulla durata, lasciando comunque la possibilità di ricominciare, prima di avere concluso la verifica, purché ci sia il tempo complessivo necessario.

Successivamente, dopo un titolo, si vede un elemento '**FORM**', che contiene in pratica le varie componenti da compilare della verifica: alcuni campi descrittivi, due gruppi di bottoni di selezione e il bottone di conclusione della verifica:

```
<FORM NAME="test_form_1" METHOD="get" ACTION="">
    ...
    <P>Cognome, nome, classe:
    <INPUT TYPE="TEXT" SIZE="60" NAME="student" VALUE="">
    </P>
    ...
    <P><STRONG>Quante sono le dita di una mano?</STRONG></P>
    <OL>
    <LI><INPUT TYPE="RADIO"
               NAME="dita_di_una_mano">0 dita</LI>
    <LI><INPUT TYPE="RADIO"
               NAME="dita_di_una_mano">1 dita</LI>
    <LI><INPUT TYPE="RADIO"
               NAME="dita_di_una_mano">2 dita</LI>
    <LI><INPUT TYPE="RADIO"
               NAME="dita_di_una_mano">3 dita</LI>
    <LI><INPUT TYPE="RADIO"
               NAME="dita_di_una_mano">4 dita</LI>
    <LI><INPUT TYPE="RADIO"
               NAME="dita_di_una_mano">5 dita</LI>
    <LI><INPUT TYPE="RADIO"
               NAME="dita_di_una_mano">6 dita</LI>
    <LI><INPUT TYPE="RADIO"
               NAME="dita_di_una_mano">7 dita</LI>
    </OL>
    ...
    <P><INPUT TYPE="button" VALUE="concludi la verifica"
        onClick="check_test_form_1(start_time)"></P>
</FORM>
```

L'elenco numerato ('**OL**') contenente gli elementi di selezione di tipo '**RADIO**' (che pertanto consentono una sola selezione nel gruppo), serve a definire la scelta di una domanda a risposta multipla.

Al bottone finale viene abbinato l'evento '**onClick**', in modo da avviare contestualmente la funzione ***check_test_form_1()***, fornendo alla stessa l'orario esatto di inizio della verifica.

Si può osservare che inizialmente, l'elemento '**FORM**' contiene anche l'attributo '**METHOD**' e '**ACTION**', che però sono perfettamente inutili, dal momento che non si usa alcun programma CGI.

Qua e là, appare anche un campo particolare, che ha lo scopo di ospitare l'indicazione del tempo a disposizione, che decresce progressivamente:

```
<P>Tempo a disposizione: <INPUT TYPE="TEXT"
NAME="countdown_2" SIZE="10" VALUE=" "></P>
```

In questo campo non si deve scrivere, ma anche se fosse, il contenuto verrebbe rimpiazzato periodicamente con l'informazione del tempo che decresce.

54.8.3 Variabili globali

All'inizio del programma appare la dichiarazione di una variabile globale:

```
<SCRIPT LANGUAGE="JavaScript" TYPE="text/javascript">
<!--
    //
    // Get the initial time stamp.
    //
    var start_time = new Date ();
```

Lo scopo è quello di accumulare al caricamento della pagina la data e l'orario iniziale. Successivamente, questa informazione viene passata come argomento della funzione ***check_test_form_1()***, come già descritto in precedenza.

54.8.4 Conto alla rovescia

Una porzione del programma JavaScript è dedicata alla gestione di un orologio che mostra il tempo a disposizione all'interno dei campi del tipo:

```
<INPUT TYPE="TEXT" NAME="countdown_2" SIZE="10" VALUE=" ">
```


Questo lavoro è svolto dalla funzione ***countdown()***, che richiede come argomento il tempo a disposizione, espresso in millesimi di secondo:

```
function countdown (time_allowed)
{
    var hours;
    var minutes;
    var seconds;
    var time;
    var time_string;
    var time_sign;
    //
    // Verify the countdown time sign.
    //
    if (time_allowed > 0)
      {
        time_sign = "";
      }
    else
      {
        time_sign = "-";
      }
    //
    // Find the absolute countdown value.
    //
    time = Math.abs (time_allowed);
    //
    hours   = (time - (time % 3600000)) / 3600000;
    time    = time - (hours * 3600000);
    minutes = (time - (time % 60000)) / 60000;
    time    = time - (minutes * 60000);
    seconds = (time - (time % 1000)) / 1000;
    //
    if (minutes < 10)
      {
        minutes = "0" + minutes;
      }
    if (seconds < 10)
      {
        seconds = "0" + seconds;
      }
    if (hours < 10)
      {
        hours = "0" + hours;
      }
    //
    time_string = time_sign + hours + ":"
                            + minutes + ":" + seconds;
    //
    // Show the time string inside some text field.
    //
    document.test_form_1.countdown_0.value = time_string;
    document.test_form_1.countdown_1.value = time_string;
    document.test_form_1.countdown_2.value = time_string;
    //
    // Reduce 10 s inside the countdown variable.
    //
    time_allowed = time_allowed - 10000;
    //
    // Wait for 10 s for the recursive call.
    //
    setTimeout ("countdown (" + time_allowed + ")", 10000);
}
```

Il lavoro iniziale della funzione è quello di separare le varie componenti del tempo, in modo da individuare le ore, i minuti e i secondi, avendo anche l'accortezza di aggiungere uno zero iniziale quando il valore corrispondente ha una sola cifra. Il risultato di questa scomposizione viene aggregato nella variabile '**time_string**', in modo da produrre qualcosa di simile a «00:10:20» (zero ore, 10 minuti e 20 secondi), avendo cura anche di mostrare un segno se il valore è negativo; fatto questo, il valore della stringa viene copiato nei campi che servono a visualizzare il conteggio ('**document.test_form_1.countdown_*n*.value**').

Una volta provveduto a mostrare il valore aggiornato nei campi rispettivi, si decrementa il tempo a disposizione di 10 s e si esegue una chiamata ricorsiva attraverso l'aiuto della funzione ***setTimeout()***.

La funzione ***setTimeout()*** serve a eseguire una funzione, passata come stringa, dopo che è trascorso un certo tempo, senza bloccare il funzionamento del programma complessivo. In questo caso, si avvia la stessa funzione ***countdown()***, passando il valore aggiornato del tempo a disposizione, dopo che sono passati 10 s. Secondo la modalità normale di programmazione, sarebbe più logico eseguire un ciclo iterativo, inserendo una pausa in quel punto, ma in questo caso ciò non è possibile, sia perché manca una funzione appropriata nel linguaggio JavaScript, sia perché una tale funzione bloccherebbe il funzionamento del programma, impedendo così la compilazione del modulo e l'invio della verifica.

Una cosa da osservare è che il ciclo di aggiornamento dell'orologio che mostra il conto alla rovescia è di 10 s, mentre sarebbe facilissimo ridurlo a un solo secondo:

```
    //
    // Reduce 1 s inside the countdown variable.
    //
    time_allowed = time_allowed - 1000;
    //
    // Wait for a second for the recursive call.
    //
    setTimeout ("countdown (" + time_allowed + ")", 1000);
```

La scelta di avere un aggiornamento meno frequente dipende solo dall'intenzione di non agitare inutilmente chi sta svolgendo la verifica.

54.8.5 Scansione di un elenco di tipo «RADIO»

Le domande a risposta multipla vengono gestite attraverso componenti del modulo di tipo '**`RADIO`**'. L'insieme dell'elenco si ottiene tramite un nome del tipo seguente; in pratica, nel caso del primo elenco si tratta del nome '**`document.test_form_1.dita_di_una_mano`**':

```
document.form.radio
```

Questo nome rappresenta precisamente un array di valori logici, con cui si può poi verificare se il bottone corrispondente è stato premuto o meno:

```
if (document.form.radio[n].checked)
  {
    …
  }
```

Il primo elemento dell'array si raggiunge con l'indice zero e corrisponde al primo dei bottoni.

Nel programma JavaScript proposto viene usata una funzione apposita per la scansione di un gruppo di bottoni di questo tipo:

```
function scan_radio_button (radio_button)
{
    var counter;
    for (counter = 0;
         counter < radio_button.length;
         counter++)
      {
        if (radio_button[counter].checked)
          {
            return (counter + 1);
          }
      }
    return 0;
}
```

In pratica, attraverso la variabile '**`radio_button`**' si ottiene l'array da scandire, quindi si utilizza un contatore per verificare i vari elementi. Se si incontra un bottone premuto, la funzione termina il suo ciclo e restituisce il valore dell'indice aumentato di una unità, mentre se non viene trovato premuto alcun bottone, viene restituito il valore zero.

Come si comprende, la chiamata di questa funzione avviene in un modo simile a quello seguente:


```
dita_di_una_mano = scan_radio_button
                   (document.test_form_1.dita_di_una_mano);
```

54.8.6 Valutazione della verifica

Si ottiene la valutazione della verifica con chiamata della funzione ***check_test_form_1()*** che richiede di fornire la data e l'ora di inizio. La chiamata di questa funzione viene innescata dal bottone che si trova alla fine del modulo:

```
<P><INPUT TYPE="button" VALUE="concludi la verifica"
        onClick="check_test_form_1(start_time)"></P>
```

La funzione in questione contiene delle informazioni fisse, che potrebbero essere passate tramite gli argomenti. Si tratta precisamente del tempo a disposizione (che deve combaciare con quello stabilito per la chiamata della funzione che esegue il conto alla rovescia) e della penalità nel punteggio da applicare a ogni minuto di ritardo:

```
function check_test_form_1 (start_time)
{
    //
    var end_time = new Date ();
    var test_time = end_time - start_time;
    var time_allowed = 1 * 60000;
    var time_delay = test_time - time_allowed;
    //
    if (time_delay < 0)
      {
        time_delay = 0;
      }
    //
    // Penalty for a minute delay.
    //
    var score_time_penalty = 0.5;
```

Oltre a questo, c'è da osservare che la funzione contiene internamente l'informazione su quali sono le risposte corrette, attraverso degli array che contengono il punteggio da dare per ogni risposta. Si osservi che in questo caso, l'elemento iniziale (con indice zero) corrisponde alla mancanza di alcuna selezione:

```
    //
    // Set the result for every choice.
    //
    var score_dita_di_una_mano = new Array();
    score_dita_di_una_mano[0] = 0;
    score_dita_di_una_mano[1] = 0;
    score_dita_di_una_mano[2] = 0;
    score_dita_di_una_mano[3] = 0;
    score_dita_di_una_mano[4] = 0;
    score_dita_di_una_mano[5] = 0;
    score_dita_di_una_mano[6] = 5;
    score_dita_di_una_mano[7] = 0;
    score_dita_di_una_mano[8] = 0;
    //
    // Set the result for every choice.
    //
    var score_dita_di_un_piede = new Array();
    score_dita_di_un_piede[0] = 0;
    score_dita_di_un_piede[1] = 0;
    score_dita_di_un_piede[2] = 0;
    score_dita_di_un_piede[3] = 0;
    score_dita_di_un_piede[4] = 0;
    score_dita_di_un_piede[5] = 0;
    score_dita_di_un_piede[6] = 5;
    score_dita_di_un_piede[7] = 0;
    score_dita_di_un_piede[8] = 0;
```

Successivamente, nella funzione vengono salvati i dati descrittivi, che servono a identificare chi ha eseguito la verifica:

```
    //
    // Save the personal data.
    //
    var student = document.test_form_1.student.value;
    var date = document.test_form_1.date.value;
```

Si passa quindi alla scansione delle risposte e al calcolo del punteggio corrispondente:

```
    //
    // Scan the radio buttons.
    //
    var dita_di_una_mano
            = scan_radio_button
                    (document.test_form_1.dita_di_una_mano);
    //
    var dita_di_un_piede
            = scan_radio_button
                    (document.test_form_1.dita_di_un_piede);
    //
    // Calculate the results.
    //
    var account_score_time_penalty;
    var account_score_dita_di_una_mano;
    var account_score_dita_di_un_piede;
    var account_score_sum = 0;
    //
    account_score_time_penalty
            = (time_delay / 1000)
               * (score_time_penalty / 60);
    //
    account_score_dita_di_una_mano
        = score_dita_di_una_mano[dita_di_una_mano];
    account_score_dita_di_un_piede
        = score_dita_di_un_piede[dita_di_un_piede];
    account_score_sum
        = account_score_sum - account_score_time_penalty;
    account_score_sum
        = account_score_sum
          + account_score_dita_di_una_mano;
    account_score_sum
        = account_score_sum
          + account_score_dita_di_un_piede;
```

Terminata l'acquisizione e valutazione dei dati, si ha cura di azzerare il modulo, in modo da assicurare che non possa essere utilizzabile il bottone per il ritorno alla pagina precedente tramite il navigatore ipertestuale per modificare la verifica già terminata:

```
    //
    // Reset the form to avoid the use of the [Back] button
    // from the browser.
    //
    document.test_form_1.reset();
```

Successivamente si passa alla fase di produzione di una nuova pagina HTML con l'esito della verifica. Si osservi che a seconda dell'interprete JavaScript usato, l'anno potrebbe essere visualizzato correttamente o meno (per esempio, l'anno 2005 potrebbe essere mostrato come 105), ma si tratta di un errore trascurabile per i fini di questo prototipo:

```
    //
    // Print a simple HTML header.
    //
    document.writeln ("<HTML>");
    document.writeln ("<HEAD>");
    document.write   ("<TITLE>");
    document.write   ("Risultato della verifica");
    document.writeln ("</TITLE>");
    document.writeln ("</HEAD>");
    document.writeln ("<BODY>");
    //
    // Print a H1 title and some personal data.
    //
    document.writeln ("<H1>Verifica banale</H1>");
    document.writeln ("<P>studente: " + student + "</P>");
    document.writeln ("<P>data: " + date + "</P>");
    //
    // Print about time.
    //
    document.write ("<P>inizio della verifica: ");
    document.write (start_time.getYear());
    document.write (".");
    document.write (start_time.getMonth() + 1);
    document.write (".");
    document.write (start_time.getDate());
    document.write (" ");
    document.write (start_time.getHours());
```



```
    document.write (":");
    document.write (start_time.getMinutes());
    document.write (".");
    document.write (start_time.getSeconds());
    document.writeln ("</P>");
    //
    document.write ("<P>conclusione della verifica: ");
    document.write (end_time.getYear());
    document.write (".");
    document.write (end_time.getMonth() + 1);
    document.write (".");
    document.write (end_time.getDate());
    document.write (" ");
    document.write (end_time.getHours());
    document.write (":");
    document.write (end_time.getMinutes());
    document.write (".");
    document.write (end_time.getSeconds());
    document.writeln ("</P>");
    //
    document.write ("<P>durata complessiva della ");
    document.write ("verifica: ");
    document.write (test_time / 1000);
    document.write (" s, ");
    document.write (test_time / 1000 / 60);
    document.write (" m;");
    document.writeln ("</P>");
    //
    document.write ("<P>tempo a disposizione: ");
    document.write (time_allowed / 1000);
    document.write (" s; ");
    document.write ("ritardo: ");
    document.write (time_delay / 1000);
    document.write (" s; ");
    document.write ("penalit&agrave;/minuto: ");
    document.write (score_time_penalty);
    document.write ("; ");
    document.write ("penalit&agrave; complessiva: ");
    document.write (account_score_time_penalty);
    document.writeln ("</P>");
    //
    // First question.
    //
    document.write ("<P>Quante sono le dita di una mano? ");
    document.write ("risposta n. ");
    document.write (dita_di_una_mano);
    document.write ("; ");
    document.write ("punteggio: ");
    document.write (account_score_dita_di_una_mano);
    document.writeln ("</P>");
    //
    // Second question.
    //
    document.write ("<P>Quante sono le dita di un piede? ");
    document.write ("risposta n. ");
    document.write (dita_di_un_piede);
    document.write ("; ");
    document.write ("punteggio: ");
    document.write (account_score_dita_di_un_piede);
    document.writeln ("</P>");
    //
    // Final result.
    //
    document.write ("<P>Punteggio complessivo della ");
    document.write ("verifica: ");
    document.write (account_score_sum);
    document.writeln ("</P>");
    //
    // End of HTML.
    //
    document.writeln ("</BODY>");
    document.writeln ("</HTML>");
```

Al termine, si può osservare l'uso della funzione ***window.print()*** per invitare alla stampa del risultato, in modo da poterlo consegnare all'insegnante. Si osservi che con alcuni navigatori potrebbe anche non funzionare, ma rimane la possibilità di comandare la richiesta di stampa tramite il menù del navigatore stesso.

Alla fine della funzione, per completezza, viene usata la funzione

window.stop(), per interrompere il caricamento della pagina, che con alcuni navigatori rimane attivo, nonostante non ci sia altro da attendere.

```
    //
    // Print with a printer.
    //
    window.print ();
    //
    // Stop loading.
    //
    window.stop ();
}
```

54.9 HTML2ps

HTML2ps[5] è un programma in grado di comporre uno o più file HTML, generando un risultato in PostScript. Questo si ottiene attraverso l'aiuto di altri programmi che devono essere installati, come per esempio TeX.

Teoricamente, HTML2ps è in grado di ricomporre assieme un documento suddiviso su più file HTML, ma questa possibilità dipende molto dall'organizzazione di questi file, all'interno dei quali, i riferimenti ipertestuali devono essere molto semplici. In generale, è possibile l'acquisizione diretta dalla rete; tuttavia, sarebbe consigliabile prima la riproduzione locale, con l'ausilio di Wget (40.8), attraverso il quale si possono modificare automaticamente i riferimenti ipertestuali, rendendo omogeneo il tutto.

HTML2ps si compone semplicemente dell'eseguibile '**`html2ps`**' (un programma scritto in Perl) e di uno o più file di configurazione. È indispensabile almeno il file di configurazione generale, '`/etc/html2psrc`', che dovrebbe essere già predisposto in modo sufficientemente buono dal sistema di installazione. Eventualmente, gli utenti possono preparare una configurazione personalizzata nel file '`~/.html2psrc`' e altri file specifici da richiamare con l'opzione '**`-f`**', oltre all'aggiunta di stili ulteriori (opzione '**`-s`**').

54.9.1 Configurazione di HTML2ps

Come accennato, la configurazione di HTML2ps è indispensabile. Di solito si predispone almeno il file di configurazione generale, '`/etc/html2psrc`', mentre gli utenti hanno la possibilità di modificare o aggiungere qualcosa attraverso il file '`~/.html2psrc`'. La sintassi per la scrittura di questi file è la stessa dei fogli di stile CSS (sezione 54.6), con l'aggiunta di un selettore specifico, '**`@html2ps`**', che serve a indicare gli aspetti particolari che riguardano HTML2ps e non possono appartenere ai fogli di stile CSS.

Bisogna tenere presente che HTML2ps è in grado di riconoscere solo una parte limitata delle dichiarazioni CSS.

HTML2ps riconosce anche i commenti CSS e le inclusioni di file di configurazione aggiuntivi, secondo la forma:

```
@include file
```

Per cominciare, è opportuno vedere un esempio abbastanza semplice di ciò che potrebbe contenere un file di configurazione, quando questo viene generato automaticamente dalla procedura di installazione.

```
/* Configurazione globale per html2ps */

@html2ps {
  package {
    ImageMagick: 1;
    PerlMagick: 1;
    TeX: 1;
    Ghostscript: 1;
    check: weblint;
    libwww-perl: 1;
    path: "/usr/bin/X11:/usr/bin";
  }
  paper {
    type: A4;
  }
  option {
    hyphenate: 0;
  }
}
```


Si può osservare che in questo esempio è stata dichiarata solo la regola corrispondente al selettore '**`@html2ps`**', all'interno della quale si trovano altre sottoregole. Generalmente, le regole tipiche di uno stile CSS si aggiungono sotto. La configurazione predefinita dello stile CSS è indicata nella pagina di manuale *html2psrc(5)* e da questa si intende quali siano le possibilità effettive di HTML2ps nel riconoscere le dichiarazioni CSS:

```
BODY {
    font-family: Times;
    font-size: 11pt;
    text-align: left;
    background: white;
}

H1, H2, H3, H4, H5, H6 {
    font-weight: bold;
    margin-top: 0.8em;
    margin-bottom: 0.5em;
}

H1 { font-size: 19pt }
H2 { font-size: 17pt }
H3 { font-size: 15pt }
H4 { font-size: 13pt }
H5 { font-size: 12pt }
H6 { font-size: 11pt }

P, OL, UL, DL, BLOCKQUOTE, PRE {
    margin-top: 1em;
    margin-bottom: 1em;
}

P {
    line-height: 1.2em;
    text-indent: 0;
}

OL, UL, DD { margin-left: 2em }

TT, KBD, PRE { font-family: Courier }

PRE { font-size: 9pt }

BLOCKQUOTE {
    margin-left: 1em;
    margin-right: 1em;
}

ADDRESS {
    margin-top: 0.5em;
    margin-bottom: 0.5em;
}

TABLE {
    margin-top: 1.3em;
    margin-bottom: 1em;
}

DEL { text-decoration: line-through }

A:link, HR { color: black }
```

54.9.1.1 Configurazione della regola corrispondente al selettore speciale @html2ps

La regola corrispondente al selettore '**`@html2ps`**' si compone di dichiarazioni e di altre sottoregole per la configurazione di HTML2ps. Nelle sezioni seguenti vengono descritti i selettori specifici di queste sottoregole.

Alcune proprietà hanno un significato booleano. A loro si assegna il valore zero per indicare *Falso* e il valore uno per indicare *Vero*.

I valori che fanno riferimento a un'unità di misura, vanno indicati come avviene nei fogli di stile CSS: il numero seguito immediatamente dall'unità di misura. La tabella 54.172 elenca le unità di misura e le sigle corrispondenti che si possono utilizzare in questa circostanza. È importante osservare che l'unica dimensione relativa riconosciuta da HTML2ps è il quadratone e non sono previste misure percentuali come invece si può fare secondo le specifiche di W3C per i fogli di stile CSS.

Tabella 54.172. Unità di misura secondo HTML2ps.

Sigla	Unità di misura
cm	Centimetri.
mm	Millimetri.
pt	Punti tipografici.
pc	Pica.
em	Quadratoni, corrispondente alla dimensione della lettera «M» maiuscola.

Tabella 54.173. Proprietà del selettore '**@html2ps**'.

Proprietà	Descrizione
`numberstyle: 0|1`	Permette di stabilire la numerazione delle pagine: zero richiede l'uso dei numeri arabi; uno corrisponde a numeri romani. Il valore predefinito per questa proprietà è il valore zero.
`showurl: 0|1`	Attivando questa proprietà booleana, si ottiene l'inserimento nella composizione dell'indirizzo URI corrispondente ai riferimenti ipertestuali. In situazioni normali questo non avviene.
`seq-number: 0|1`	Permette di abilitare la numerazione dei titoli '**H1**', '**H2**',... '**H6**'. In condizioni normali, questo non avviene.

La sottoregola '**package**' serve a definire la disponibilità o meno di altri programmi di cui HTML2ps potrebbe avere bisogno. Di conseguenza si tratta di assegnamenti di valori booleani, dove zero rappresenta l'assenza del programma in questione e in generale è anche il valore predefinito.

```
@html2ps {
  …
  package {
    proprietà
    …
  }
  …
}
```

Tabella 54.174. Proprietà della sottoregola '**package**'.

Proprietà	Descrizione
`PerlMagick: 0|1`	Indica la mancanza o la disponibilità di PerlMagick.
`ImageMagick: 0|1`	Indica la mancanza o la disponibilità di ImageMagick.
`Ghostscript: 0|1`	Indica la mancanza o la disponibilità di Ghostscript.
`TeX: 0|1`	Indica la mancanza o la disponibilità di TeX.
`dvips: 0|1`	Indica la mancanza o la disponibilità di '**dvips**'.
`libwww-perl: 0|1`	Indica la mancanza o la disponibilità del modulo Perl Libwww-Perl.
`path:` ***percorsi_aggiuntivi***	Si tratta dell'indicazione di percorsi aggiuntivi per la ricerca degli eseguibili. Serve a garantire che i programmi utilizzati da HTML2ps siano raggiungibili per tutti gli utenti. In generale, in presenza di un sistema configurato bene, non dovrebbe essere necessaria l'indicazione di questa dichiarazione.

La sottoregola '**paper**' serve a definire le caratteristiche della carta. In generale si tratta solo delle dimensioni.

```
@html2ps {
  …
  paper {
    proprietà
    …
  }
  …
}
```

Tabella 54.175. Proprietà della sottoregola '**paper**'.

Proprietà	Descrizione
`type:` ***tipo_di_carta***	La direttiva serve a definire le dimensioni della carta, attraverso l'indicazione di un nome standard; per esempio: '**A0**', '**A1**', ... '**A10**', '**B0**', '**B1**', ... '**B10**', '**letter**', '**legal**', ecc. In alternativa, si possono indicare le dimensioni precise attraverso le proprietà '**height**' e '**width**'.
`height:` ***dimensione_assoluta***	Permette di definire l'altezza del foglio.
`width:` ***dimensione_assoluta***	Permette di definire la larghezza del foglio.

La sottoregola '**option**' serve a definire l'utilizzo di alcune opzioni, a cui si può accedere anche attraverso la riga di comando. Vengono descritte prima le dichiarazioni da indicare nel file di configurazione e poi le opzioni corrispondenti della riga di comando.

```
@html2ps {
  …
  option {
    proprietà
    …
  }
  …
}
```

Tabella 54.176. Proprietà della sottoregola '**option**' e opzioni corrispondenti della riga di comando.

Proprietà e opzione della riga di comando	Descrizione

`twoup: 0|1`
`-2`
`--twoup`
	Se attivato, fa in modo di ottenere un testo organizzato su due colonne verticali.

`toc: {f|h|t}[b]` `-C {f|h|t}[b]`	Fa in modo che venga generato un indice generale, in base alle opzioni specificate da una o più lettere: la lettera '**b**' richiede che l'indice generale sia collocato all'inizio; la lettera '**f**' richiede che l'indice generale sia generato a partire dai riferimenti contenuti nel documento; la lettera '**h**' richiede che l'indice generale sia generato a partire dai titoli definiti dagli elementi HTML da '**H1**' a '**H6**'; la lettera '**t**' richiede che l'indice generale sia generato a partire da elementi '**LINK**' contenenti l'attributo '**REV=TOC**'.

Proprietà e opzione della riga di comando	Descrizione

`DSC: 0|1`
`-D`
`--DSC`
	Se attivato, fa in modo di generare un file PostScript aderente alle specifiche DSC. In generale, per ottenere un file PostScript completo, è necessario attivare questa opzione.

`encoding:` ***codifica*** `-e` ***codifica*** `--encoding` ***codifica***	Permette di definire la codifica in cui è realizzato il file HTML. Il valore predefinito è '**`ISO-8859-1`**', ma sono poche altre le possibilità (si deve consultare la pagina di manuale).

`hyphenate: 0|1`
`-H`
`--hyphenate`
	Se attivato, fa in modo che il testo possa essere separato in sillabe, per facilitare l'impaginazione.

`language:` ***linguaggio*** `-l` ***linguaggio*** `--language` ***linguaggio***	Permette di indicare un linguaggio diverso da quello che può essere stato dichiarato nell'elemento '**`BODY`**' con l'attributo '**`LANG`**' di un documento HTML. La stringa che definisce il linguaggio va scelta in base a quanto già consentito dall'HTML (tabella 13.4).

`landscape: 0|1`
`-L`
`--landscape`
	e attivato, genera pagine orientate in modo orizzontale.

`number: 0|1`
`-n`
`--number`
	e attivato, fa in modo di aggiungere i numeri di pagina.

`startno:` ***n*** `-N` ***n*** `--startno` ***n***	Specifica il numero iniziale delle pagine. Il valore predefinito è uno.

`xref: 0|1`
`-R`
`--xref`
	Se attivato, fa in modo di aggiungere dei riferimenti visivi nel testo, in corrispondenza di quelli ipertestuali contenuti nel documento HTML.

`scaledoc:` ***scala_percentuale*** `-s` ***scala_percentuale*** `--scaledoc` ***scala_percentuale***	Riduce o amplia la scala del documento: il valore unitario rappresenta la situazione normale, di una scala pari al 100 %; valori superiori indicano un ingrandimento, mentre valori inferiori indicano una riduzione (si usa il punto per separare la parte intera dalle cifre decimali).

Proprietà e opzione della riga di comando	Descrizione

`web: {a|b|l|r|s}[p|L|`***n***`]`
`-W {a|b|l|r|s}[p|L|`***n***`]`
`-web {a|b|l|r|s}[p|L|`***n***`]`
	Fa in modo che vengano utilizzati più file HTML che si ritiene facciano parte dello stesso documento. Il modo in cui vengono presi in considerazione questi file dipende dalla stringa composta nel modo mostrato dallo schema sintattico.
	Si utilizza la lettera '**`a`**' per seguire tutti i riferimenti ipertestuali; la lettera '**`b`**' per seguire soltanto i riferimenti ipertestuali che riguardano la stessa directory del file iniziale; la lettera '**`l`**' per seguire soltanto i riferimenti ipertestuali che contengono l'attributo '**`REL=NEXT`**' all'interno dell'elemento '**`LINK`**'; la lettera '**`r`**' per seguire soltanto i riferimenti ipertestuali relativi; la lettera '**`s`**' per seguire solo i riferimenti allo stesso nodo del documento di partenza.
	Inoltre, la lettera '**`p`**' fa in modo che sia chiesta conferma per ogni file HTML da aggiungere (ciò avviene in ogni caso quando si superano i 50 file); la lettera '**`L`**' serve a riordinare i documenti in base alla struttura gerarchica; un numero intero ('***n***') un numero indica il livello massimo di ricorsione, tenendo conto che il valore predefinito è di quattro livelli.

La sottoregola '**`margin`**' permette di definire esplicitamente i margini della pagina.

```
@html2ps {
  …
  margin {
    proprietà
    …
  }
  …
}
```

Questa sottoregola è superata e viene sostituita dalla configurazione nel file di stile CSS, utilizzando la regola '**`@page`**', introdotta dalle specifiche CSS2.

Tabella 54.177. Proprietà della sottoregola superata '**`margin`**'.

Proprietà	Descrizione
`left` ***margine_sinistro*** `right` ***margine_destro***	Indicano i margini sinistro e destro rispettivamente. Il valore predefinito è '**`2.5cm`**', pari a 2,5 cm.
`top` ***margine_superiore*** `bottom` ***margine_inferiore***	Indicano i margini superiore e inferiore rispettivamente. Il valore predefinito è '**`3cm`**', pari a 3 cm.
`middle` ***distanza_tra_colonne***	Indica la distanza orizzontale tra le colonne, quando si stampano due colonne per pagina. Il valore predefinito è '**`2cm`**', pari a 2 cm.

La sottoregola '**`xref`**' permette di definire esplicitamente il modo in cui vengono indicati i riferimenti nel testo, quando questa funzionalità è stata abilitata.

```
@html2ps {
  …
  xref {
    proprietà
    …
  }
  …
}
```

Tabella 54.178. Proprietà della sottoregola '**xref**'.

Proprietà	Descrizione
`text:` ***modello***	Permette di definire il modello da utilizzare, tenendo conto che il simbolo '**$N**' viene rimpiazzato con il numero della pagina. Il modello predefinito è '**[p $N]**'.
`passes:` ***n***	Permette di definire il numero di passaggi necessario per determinare in modo corretto i riferimenti incrociati. Il valore predefinito è il valore uno, ma l'inserzione del testo corrispondente al modello potrebbe cambiare la sequenza delle pagine, per cui si potrebbe rendere necessario un numero maggiore di passaggi.

La sottoregola '**quote**' permette di definire esplicitamente l'uso delle virgolette più appropriate in base al linguaggio. Queste virgolette vengono inserite nel testo in corrispondenza degli elementi '**Q**'. In generale, i valori predefiniti per la lingua italiana sono già corretti. Viene mostrato solo un esempio per comprendere intuitivamente come si potrebbe adoperare questa sottoregola:

```
quote {
  it {
    open: "«";
    close: "»";
    open2: "‘";
    close2: "’";
  }
}
```

Si intende dall'esempio che sono disponibili solo due livelli di virgolette.

La sottoregola '**toc**' permette di definire alcune caratteristiche relative all'indice generale, quando la sua realizzazione è stata richiesta espressamente. In particolare si può utilizzare la proprietà '**level**' alla quale si assegna un numero, che sta a indicare i livelli da prendere in considerazione. Il valore predefinito è sei, che produce una voce per ogni tipo di titolo '**Hn**' (da '**H1**' a '**H6**').

La sottoregola '**hyphenation**' permette di definire la collocazione del file TeX contenente i modelli per la separazione in sillabe. La cosa si fa distinguendo tra diversi linguaggi. L'esempio seguente dovrebbe essere sufficiente a intendere intuitivamente la cosa:

```
hyphenation {
  it {
    file: "/usr/share/texmf/tex/generic/hyphen/ithyph.tex";
  }
  en {
    file: "/usr/share/texmf/tex/generic/hyphen/ushyph1.tex";
  }
}
```

Le sottoregole '**header**' e '**footer**' permettono di definire l'intestazione e il fondo pagina, dove di solito si collocano alcune informazioni ricorrenti assieme al numero della pagina. Le proprietà di queste sottoregole sono praticamente le stesse; qui vengono elencate solo alcune di queste proprietà nella tabella 54.182. La tabella 54.181 elenca alcuni simboli che possono essere utilizzati per definire i modelli delle intestazioni e dei fondo pagina.

Tabella 54.181. Simboli utilizzabili nelle intestazioni e nei fondo pagina.

Simbolo	Corrispondenza
`$T`	Titolo del documento.
`$A`	Autore, come specificato in '**<META NAME="Author" CONTENT="…">**'.
`$U`	URI del documento.
`$N`	Numero di pagina.
`$H`	Titolo attuale ('**H1**'...'**H3**').
`$D`	Data e orario attuale.
`\$`	Dollaro.

Tabella 54.182. Alcune proprietà utilizzabili nell'intestazione e nel fondo delle pagine.

Proprietà	Contenuto
`left`	Intestazione allineata a sinistra.
`center`	Intestazione al centro.
`right`	Intestazione allineata a destra.
`odd-left`	Intestazione delle pagine dispari allineata a sinistra.
`odd-center`	Intestazione delle pagine dispari al centro.
`odd-right`	Intestazione delle pagine dispari allineata a destra.
`even-left`	Intestazione delle pagine pari allineata a sinistra.
`even-center`	Intestazione delle pagine pari al centro.
`even-right`	Intestazione delle pagine pari allineata a destra.
`font-family`	Tipo di carattere da usare (predefinito Helvetica).
`font-size`	Dimensione del carattere (predefinito 8 punti).
`font-style`	Forma del carattere (predefinita la forma normale).
`font-weight`	Spessore del carattere (predefinito lo spessore normale).

54.9.1.2 Configurazione in cascata

La configurazione di HTML2ps segue la logica dei fogli di stile CSS, anche per ciò che riguarda la sua definizione in cascata. In generale: il file '`/etc/html2psrc`' contiene le indicazioni essenziali; il file '`~/.html2psrc`' contiene la configurazione personalizzata; l'opzione '**-f**' consente di aggiungere altra configurazione specifica; l'opzione '**-S**' consente di aggiungere una stringa ulteriore allo stile.

Quando si utilizza l'opzione '**-f**', se si vuole evitare di eliminare la configurazione standard dei file '`/etc/html2psrc`' e '`~/.html2psrc`', si deve iniziare con i due punti ('**:**'), come si vede nell'esempio seguente:

```
$ html2ps -f :locale manuale.html > manuale.ps [Invio]
```

Si possono anche sommare assieme più configurazioni o stili CSS locali, come si vede nell'esempio seguente, dove si utilizzano i file '`locale`', '`A4`' e '`numerato`':

```
$ html2ps -f :locale:A4:numerato ↩
↪        manuale.html > manuale.ps [Invio]
```

L'opzione '**-S**' serve solo per aggiungere una regola al volo, indicandola direttamente nella riga di comando, come si vede nell'esempio seguente:

```
$ html2ps -f :locale -S "H1 { color: blue }" ↩
↪        manuale.html > manuale.ps [Invio]
```

54.9.2 Avvio di HTML2ps

HTML2ps si utilizza attraverso l'eseguibile '**html2ps**', con la sintassi seguente:

```
html2ps opzioni [file_html]
```

Il file da convertire può essere indicato nella riga di comando, dove in tal caso può trattarsi anche di un URI, oppure può essere fornito attraverso lo standard input.

Quasi tutte le opzioni di questo programma sono richiamabili anche tramite una proprietà corrispondente nella sottoregola '**option**', come è già stato descritto. Qui vengono riepilogate le opzioni più importanti nella tabella 54.183. In particolare, si può osservare che si può indicare il nome del file da generare attraverso l'opzione '**-o**', oppure '**--output**', altrimenti il risultato della conversione viene emesso attraverso lo standard output.

Tabella 54.183. Riepilogo delle opzioni più comuni.

Opzione	Descrizione
`-2` `--twoup`	Due colonne verticali.
`-D` `--DSC`	Genera un file PostScript DSC (standard).
`-e` `--encoding`	Stabilisce la codifica originale.
`-H` `--hyphenate`	Abilita la separazione in sillabe.
`-L` `--landscape`	Orientamento orizzontale.
`-n` `--number`	Aggiunge i numeri alle pagine.
`-o` `--output`	Specifica il file PostScript da generare.
`-R` `--xref`	Mostra gli URI dei riferimenti ipertestuali.
`-s` `--scaledoc`	Cambia la scala del documento.
`-W` `--web`	Definisce come gestire più file HTML assieme.
`-f` `--rcfile`	Specifica i file di configurazione aggiuntivi o alternativi.
`-S` `--style`	Specifica una regola aggiuntiva al volo.

Segue la descrizione di alcuni esempi.

- `$ html2ps -o documento.ps documento.html` [*Invio*]

  Converte il file '`documento.html`' nel file '`documento.ps`'.

- `$ html2ps -2 -o documento.ps documento.html` [*Invio*]

  Converte il file '`documento.html`' nel file '`documento.ps`', che risulta organizzato in due colonne verticali.

- `$ html2ps -R -o documento.ps documento.html` [*Invio*]

  Converte il file '`documento.html`' nel file '`documento.ps`', che contiene dei riferimenti incrociati visibili.

- `$ html2ps -2 -s 0.5 -o documento.ps documento.html` [*Invio*]

  Converte il file '`documento.html`' nel file '`documento.ps`', che risulta organizzato in due colonne verticali, con la dimensione del carattere ridotta alla metà.

- `$ html2ps -W b -o XFree86.ps ↩`
  `↪ XFree86-Video-Timings-HOWTO.html` [*Invio*]

  Converte i file HTML che iniziano da '`XFree86-Video-Timings-HOWTO.html`' in un solo file PostScript, denominato '`XFree86.ps`'. In particolare viene richiesto di seguire solo i riferimenti ipertestuali rivolti alla stessa directory di partenza.

54.9.3 Particolarità nell'HTML

HTML2ps interpreta alcuni «comandi» speciali all'interno del file HTML. Si tratta di:

- salto pagina incondizionato, il quale si ottiene con uno dei comandi seguenti:

```
<HR class=PAGE-BREAK>
```

```
<?page-break>
```

```
<!--NewPage-->
```

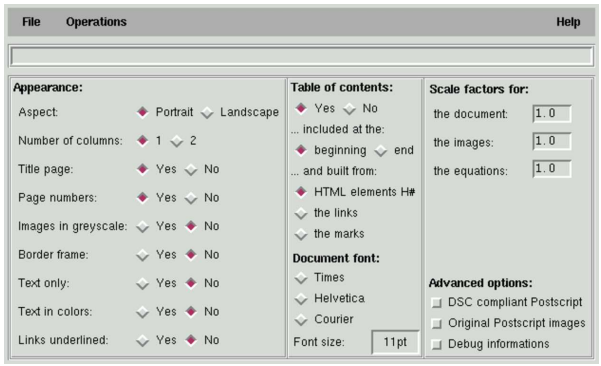
- esclusione di parte del testo dalla composizione stampata, attraverso un elemento '**DIV**' speciale:

```
<DIV class=NOPRINT>
    ...
    <!-- Testo che viene ignorato da HTML2ps -->
    ...
</DIV>
```

54.9.4 Programma frontale per semplificare l'utilizzo di HTML2ps

Assieme a HTML2ps si dovrebbe trovare un programma aggiuntivo che facilita il suo utilizzo attraverso un pannello grafico. Si tratta dell'eseguibile '**xhtml2ps**', che si vede in particolare nella figura 54.188. Il suo utilizzo dovrebbe essere intuitivo, dal momento che si rifà alle opzioni delle riga di comando.

Figura 54.188. Programma frontale per il controllo di HTML2ps.



54.10 Introduzione a Amaya

Amaya[6] è un sistema visuale integrato di navigazione e composizione di documenti HTML e XHTML. È interessante notare che Amaya è anche in grado di riconoscere e utilizzare i fogli di stile CSS.

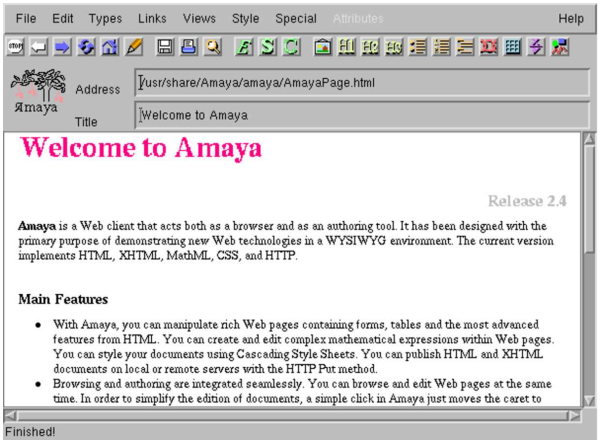
Amaya è disponibile per vari sistemi operativi. Probabilmente, questo fatto ha spinto gli sviluppatori del programma a costruire per lui un mondo a parte. In particolare, la tastiera viene gestita da Amaya in modo indipendente dal sistema sottostante.

L'avvio di Amaya è molto semplice, attraverso l'eseguibile '**amaya**', dal momento che gli argomenti sono tutti facoltativi:

```
amaya [-display schermo] [file | uri]
```

La figura 54.189 mostra come si presenta all'avvio, quando non si indica alcun file.

Figura 54.189. Amaya.



Amaya è un sistema di composizione HTML e XHTML, molto sofisticato e molto serio nel suo approccio a questi formati. Qui si introduce semplicemente il suo utilizzo, tenendo conto che la documentazione originale, accessibile anche dal menù *Help*, è buona.

54.10.1 Navigazione e composizione

Amaya è sia un navigatore HTTP, sia un sistema di composizione in HTML. Questo fatto ha delle implicazioni nel suo utilizzo che a prima vista possono sembrare un po' strane, benché siano assolutamente logiche. Per prima cosa è importante sapere che è possibile controllare la modalità di accesso al documento, attraverso la voce *Editor Mode* del menù *Edit*. Attivandola si abilita la modifica del documento; disattivandola si richiede espressamente di accedere in sola lettura.

Quando Amaya accede in sola lettura, si comporta come un navigatore normale; quando è consentita la modifica, il documento può essere alterato e salvato successivamente.

Quando si accede a un riferimento ipertestuale, come si fa di solito con i navigatori, il documento che si ottiene può occupare la stessa finestra di partenza, oppure può essere messo in un'altra. La scelta è abbastanza logica: se il documento di partenza non è stato alterato, si utilizza la stessa finestra iniziale.

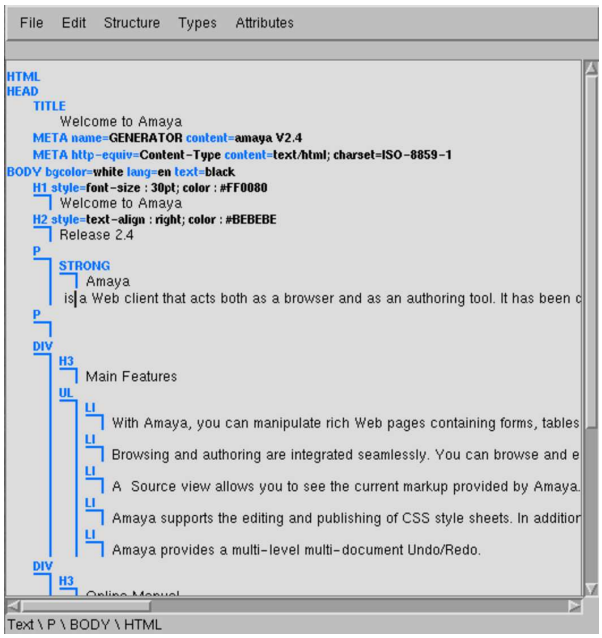
Per selezionare un riferimento ipertestuale, in condizioni normali serve un clic doppio con il primo tasto del mouse, perché con uno solo si posiziona semplicemente il cursore del testo. È possibile modificare la configurazione per fare in modo che basti un solo clic, ma in generale questa non è una buona idea, dal momento che diventerebbe difficile portare il cursore sopra un riferimento ipertestuale.

54.10.1.1 Modifica del documento

La modifica di un documento HTML può avvenire in modo visuale, diretto, attraverso la finestra che si usa anche per la sua lettura. La vera «forza» di Amaya sta nella possibilità di accedere al documento in una forma diversa, attraverso la sua struttura, in modo da avere una visione più chiara di ciò che si sta facendo.

Dal menù *Views* si possono selezionare le voci *Show structure* e *Show alternate*. La prima apre una finestra separata contenente la struttura, come si vede nell'esempio di figura 54.190, la seconda mostra il documento in un modo alternativo, precisamente in forma testuale senza grafica. La modifica in una di queste finestre si ripercuote simultaneamente su tutte le altre.

Figura 54.190. La visione della struttura.



Dallo stesso menù è possibile selezionare la voce *Show source* per accedere a una finestra contenente il sorgente del documento. Anche se è possibile modificare il testo direttamente nel sorgente, le modifiche non si applicano istantaneamente alle altre finestre, a meno di utilizzare la voce *Synchronize* dal menù *File*. Tuttavia, lo svantaggio nell'accedere direttamente al sorgente sta nel fatto che Amaya ha difficoltà a correggere gli errori nell'uso dell'HTML da parte di un autore inesperto, mentre nelle altre finestre questo non può avvenire, perché la struttura è sotto il pieno controllo del programma.

È interessante notare che alla base di ogni finestra utile per accedere alla modifica del documento appare l'indicazione sintetica della struttura del punto in cui si trova il cursore. Per esempio, la sequenza

```
Text \ P \ BODY \ HTML
```

indica che si tratta di testo contenuto in un elemento '**P**', che è contenuto nell'elemento '**BODY**', che a sua volta è parte dell'elemento '**HTML**':

```
<HTML>
    <BODY>
        <P>... <!-- Testo --> </P>
    </BODY>
</HTML>
```

Oltre alle specificità di Amaya, il suo funzionamento è abbastanza intuitivo. Si comprende che per poter essere utilizzato in modo conveniente, è più importante conoscere bene le potenzialità dell'HTML e dei fogli di stile CSS, prima di cercare di approfondire l'uso di questo programma.

54.10.2 Configurazione

La maggior parte della configurazione di Amaya è accessibile attraverso una delle voci del sottomenù *Preferences* del menù *Special*.

Nella directory personale dell'utente che utilizza il programma, Amaya crea la sottodirectory '`.amaya/`', in cui inserisce il file di configurazione generale '`thot.rc`', la sottodirectory per la sua memoria cache, '`libwww-cache/`' e i propri file temporanei. A parte la collocazione del file '`thot.rc`', il resto può essere spostato altrove attraverso la configurazione.

54.10.3 Aggregazione di un documento composto

Amaya è in grado di aggregare un documento composto da più «pagine» HTML in un solo file, attraverso la voce *Make book* del menù *Special*.

Per ottenere questo risultato si parte da un file HTML composto da un titolo contenuto in un elemento '**H1**', seguito da testo e da una serie di riferimenti. Questi riferimenti (l'elemento '**A**' con l'attributo '**HREF**') sono organizzati solitamente in un elenco puntato o numerato, ma a parte questo, tali riferimenti devono contenere anche l'attributo '**REL**', a cui viene assegnato il valore '**chapter**' o '**subdocument**'. L'esempio seguente rappresenta bene questa struttura di partenza:

```
<!DOCTYPE HTML PUBLIC
    "-//W3C//DTD HTML 4.0 Transitional//EN"
    "http://www.w3.org/TR/REC-html40/loose.dtd">
<html>
<head>
  <title>Using Amaya</title>
  <style type="text/css">
    BODY { background-color : #FFFFFF }
  </style>
</head>
<body lang="en">
<!-- ... -->
<h1 style="text-align : center">Using Amaya</h1>
<!-- ... -->
<p>Each following section gives a short description of how
to use a specific Amaya functionality.</p>
<ul>
  <li><a rel="Chapter"
  href="Browsing.html#Browsing">Browsing with Amaya</a></li>
  <li><a rel="Chapter"
  href="Selecting.html#Selecting">Selecting</a></li>
  <li><a rel="Chapter"
  href="Searching.html#Searching">Searching and replacing
  text</a></li>
  <li><a rel="Chapter" href="Views.html#Views">Displaying
  Views</a></li>
  <li><a rel="Chapter"
  href="Creating.html#Creating">Creating new
  elements</a></li>
<!-- ... -->
</ul>
<!-- ... -->
<p><a name="There">There is also a brief introduction </a>
which explains some of the different types that can be used
in Amaya such as headings, lists, and quotations, and how to
use them.</p>
<ul>
  <li><a href="HTML-elements/infoTypes.html"
  rel="Chapter">Information types in HTML</a></li>
  <li><a href="HTML-elements/structure.html"
  rel="Chapter">HTML Document Structure</a></li>
  <li><a href="HTML-elements/headings.html"
  rel="Chapter">Headers</a></li>
<!-- ... -->
</ul>
<hr>
<!-- ... -->
</body>
</html>
```

In particolare, l'elemento

```
<a href="HTML-elements/structure.html" rel="Chapter">HTML
Document Structure</a>
```

implica l'inclusione del corpo del file '`HTML-elements/structure.html`' in quel punto, al posto del suo riferimento.

Per la precisione, si possono distinguere questi casi: quando il riferimento è fatto a un documento completo, come appena visto, si ottiene l'inclusione del contenuto del suo elemento '**BODY**'; se invece il riferimento è fatto a un'etichetta di un certo elemento, viene incorporato solo il contenuto di quell'elemento.

Nella realizzazione di un documento articolato in più file differenti, converrebbe avere l'accortezza di delimitare la parte sostanziale del testo di ogni file HTML in un elemento '**DIV**' provvisto di etichetta a cui poter fare riferimento attraverso l'indice di partenza (l'attributo '**ID**'). In questo modo si potrebbero escludere dall'aggregazione una serie di informazioni che servono solo per la navigazione (pulsanti per avanzare, indietreggiare o raggiungere l'indice).

Un indice di partenza può anche fare riferimento a file che contengono a loro volta dei sottoindici, realizzando quindi una struttura ad albero abbastanza articolata. Amaya continua ad aggregare i file finché trova elementi '**A**' contenenti l'attributo '**REL**' a cui sono assegnate le parole chiave già indicate.

54.11 HTMLDOC

HTMLDOC[7] è un sistema di composizione basato su HTML. In pratica, si parte da uno o più file HTML e si ottiene una composizione in PostScript, PDF e HTML. I file HTML di partenza devono avere una struttura ragionevolmente semplice, dove il testo sia strutturato in capitoli, inizianti con un'intestazione '**H1**', suddivisi nel modo consueto, attraverso le intestazioni di livello inferiore ('**H2**', '**H3**',... '**H7**'). In questo modo, HTMLDOC è in grado di generare automaticamente un indice generale e diventa utile tutto il sistema anche quando l'obiettivo è la generazione di una composizione finale nello stesso formato HTML.

HTMLDOC è disponibile sia su piattaforme Unix, sia su sistemi MS-Windows. In particolare, può funzionare in modo interattivo, attraverso l'interfaccia grafica, oppure in modo non interattivo utilizzando semplicemente opzioni della riga di comando.

54.11.1 Sorgente HTML

Il sorgente di un documento HTMLDOC può essere contenuto in un file singolo, oppure in più file HTML distinti. La prima cosa che dovrebbe apparire nel corpo del file HTML è un'intestazione di tipo '**H1**'.

In questi file HTML non sono ammissibili le inserzioni di oggetti, a parte le immagini nel modo consueto, attraverso l'elemento '**IMG**'; non si possono usare le cornici (*frame*); vengono ignorati gli script e le applet. È possibile selezionare soltanto i tipi di carattere standard, corrispondenti a: Helvetica, Times e Courier. Dal momento che HTMLDOC è disponibile anche su piattaforma MS-Windows, il carattere Arial viene convertito automaticamente in Helvetica. Non sono gestiti i fogli di stile e le tabelle sono riconosciute solo al livello di HTML 3.2.

La documentazione di HTMLDOC è scritta in questo modo, per cui può essere osservata la tecnica utilizzata e il risultato che si ottiene. In generale, dopo l'elemento '**HEAD**', all'interno dell'elemento '**BODY**' si comincia subito con un'intestazione '**H1**'. L'esempio seguente è un estratto del file '`intro.html`', che è il primo del gruppo di file che compongono la documentazione di HTMLDOC:

```
<HTML>
<HEAD>
   <META HTTP-EQUIV="Content-Type"
         CONTENT="text/html; charset=iso-8859-1">
   <META NAME="author" CONTENT="Michael R. Sweet">
   <META NAME="copyright"
         CONTENT="Copyright 1997-1999, See the GNU General
                  Public License for Details.">
   <META NAME="Author"
         CONTENT="Michael R. Sweet, Easy Software Products">
   <TITLE>HTMLDOC 1.7 User's Guide</TITLE>
</HEAD>
<BODY>

<DIV ALIGN=right>
<H1>
Introduction</H1></DIV>

<H2>About This Software</H2>
This document describes how to use the <I>HTMLDOC</I>
software, version 1.7. <I>HTMLDOC</I> is a HTML document
processing program that generates indexed HTML, Adobe&reg;
PostScript<SUP>TM</SUP>, and Adobe Portable Document Format
```

```
(PDF 1.2) files suitable for printing or online viewing.

<P>No restrictions are placed upon the output produced by
<I>HTMLDOC</I>.

<H2>History</H2>

Like many programs <I>HTMLDOC</I> was developed in response
to a need my company had for generating high-quality
documentation in printed and
...
...

<H2>Why Just HTML?</H2>

Some people have asked why this program only deals with HTML
input files and is not able to read any Standard Generalized
Markup Language (SGML) file. The reasons are numerous but
basically boil down to:
...
...
</BODY>
</HTML>
```

Eventualmente è possibile convertire un file HTML singolo che non sia conforme a questa struttura, utilizzando un'opzione apposita, '**--webpage**', con la quale non si ottiene più l'indice generale, ma soprattutto non è possibile aggregare più file HTML assieme in un documento finale unico.

HTMLDOC è progettato per gestire documenti di dimensioni molto grandi; tuttavia esistono dei limiti, fissati nel file '`config.h`', che appartiene ai sorgenti del programma. Eventualmente si possono estendere tali limiti modificando questo file e ricompilando successivamente i sorgenti.

54.11.2 Funzionamento

HTMLDOC si compone dell'eseguibile '**htmldoc**', che può essere avviato senza argomenti, per ottenere un funzionamento interattivo:

```
htmldoc [opzioni file_html...]
```

È prevista la presenza di un file di configurazione personale, dei singoli utenti. Si tratta del file '`~/.htmldocrc`'. Questo file viene creato la prima volta dall'eseguibile '**htmldoc**' e potrebbe essere utile ritoccare la direttiva di dichiarazione del programma usato per modificare i file HTML sorgenti:

```
#HTMLDOCRC 1.7
EDITOR=amaya %s
```

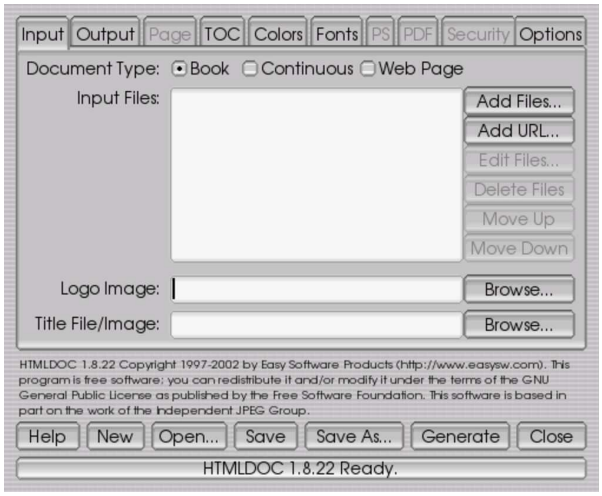
In questo caso, si fa in modo di utilizzare Amaya per la modifica del sorgente HTML, quando questo tipo di programma viene richiamato da HTMLDOC.

Purtroppo non è disponibile una documentazione sufficiente sulle direttive di questo file di configurazione.

Nel seguito viene mostrato il funzionamento interattivo di HTMLDOC, abbinando anche la descrizione delle opzioni che possono servire per ottenere lo stesso risultato senza interazione con il programma. Intanto, nella figura 54.197 si può osservare come appare inizialmente, dopo l'avvio con il comando seguente:

```
$ htmldoc [ Invio ]
```


Figura 54.197. Aspetto iniziale di HTMLDOC, quando viene avviato senza argomenti.



Nella parte bassa del pannellino grafico, appaiono alcuni pulsanti grafici, che fanno riferimento alla possibilità di creare e salvare un file contenente tutte le informazioni sulla composizione che si vuole generare. Questo file può essere inteso come la configurazione del libro che si vuole comporre.

- Il pulsante NEW serve a eliminare il lavoro in corso, per ricominciare con un altro nuovo.
- Il pulsante OPEN permettere di accedere al file system per selezionare un file contenente le informazioni su una composizione già predisposta in precedenza.
- I pulsanti SAVE e SAVE AS consentono di salvare l'impostazione attuale, eventualmente dando un nome nuovo alla cosa.
- Il pulsante GENERATE avvia la composizione in base alle informazioni indicate, generando uno o più file PostScript, PDF o HTML, a seconda di quanto specificato a questo proposito.
- Il pulsante CLOSE termina il funzionamento di HTMLDOC.

La parte centrale del pannello grafico di HTMLDOC cambia in funzione del lembo superiore selezionato.

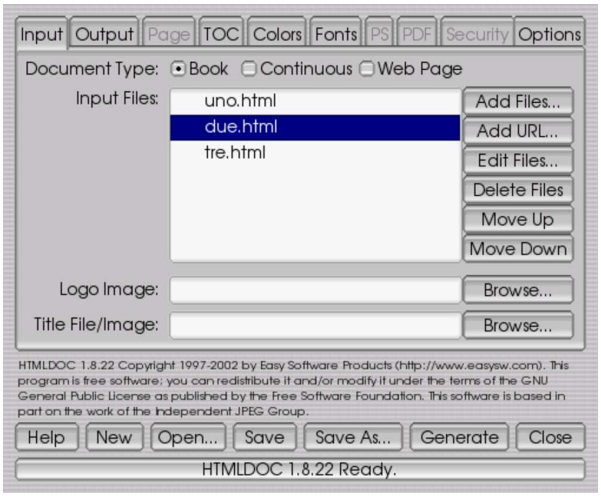
54.11.3 Definizione dei file sorgenti

La prima fase per arrivare alla composizione è quella della selezione dei file HTML che compongono i sorgenti. Ciò si fa dalla finestra che si ottiene selezionando l'etichetta *Input*, che corrisponde alla situazione in cui si presenta HTMLDOC all'avvio.

In alto, si può osservare la presenza di due pulsanti di selezione, dove si può specificare il tipo di sorgente che si utilizza. La voce BOOK indica l'intenzione di utilizzare uno o più file HTML per generare un documento unico, in forma di libro, mentre la voce WEB PAGE, corrispondente all'opzione '**--webpage**', specifica che si tratta di un solo file HTML che non ha la struttura richiesta per realizzare un libro.

Il pulsante grafico laterale ADD FILE consente di accedere al file system per selezionare i file HTML che compongono i sorgenti del documento che si vuole comporre. Nella figura 54.198 appaiono selezionati i file '`uno.html`', '`due.html`' e '`tre.html`'.

Figura 54.198. Sono stati selezionati tre file.



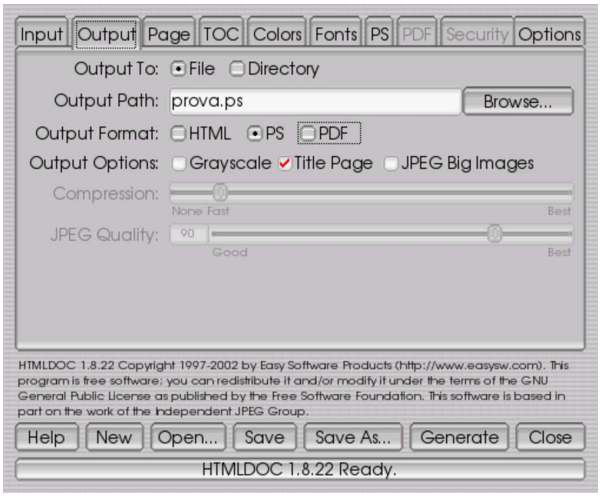
Sul lato destro si possono osservare anche altri pulsanti grafici, che si abilitano solo quando si seleziona uno o più nomi nell'elenco di quelli che compongono l'insieme dei sorgenti del documento: EDIT FILE permette di avviare automaticamente il programma per la modifica dei file HTML; DELETE FILE elimina i nomi evidenziati dall'elenco, ma senza cancellarli dal file system; MOVE UP e MOVE DOWN consentono di spostare il nome selezionato in alto o in basso, cambiando l'ordine.

Per completare l'estetica della composizione, è possibile specificare un'immagine da usare come logo e un'altra immagine da utilizzare nella copertina (ammesso che sia stata ablitata la sua generazione). Nel primo caso si può usare l'opzione '**`--logoimage`** ***file_immagine***'; nel secondo l'opzione '**`--titlefile`** ***file_immagine***'.

54.11.4 Composizione

L'etichetta *Output* consente di accedere alla definizione del file o dei file che si vogliono ottenere dalla composizione. Il risultato della composizione può essere un file oppure una directory, selezionando FILE oppure DIRECTORY, rispettivamente. Queste due voci corrispondono alle opzioni '**`--outfile`** ***file***' e '**`--outdir`** ***directory***'. Il file o la directory in questione si indicano sotto, mentre più giù si specifica esattamente il tipo di composizione che si vuole generare: HTML, PS, PS2 e PDF (opzione '**`--format {ps1|ps2|pdf|html}`**').

Figura 54.199. Definizione del risultato della composizione che si vuole ottenere.



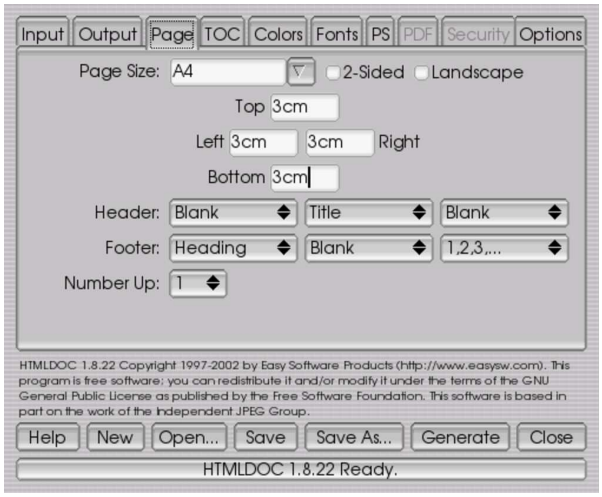
Il senso delle opzioni che appaiono nel resto della maschera è abbastanza intuitivo. Si può osservare il fatto che sia possibile cambiare il colore dello sfondo e anche utilizzare un'immagine per questo.

Se si indica una directory, si intende generare un gruppo di file nella directory stessa. Ciò può essere utile nella composizione in HTML, ma funziona nello stesso modo anche per le altre forme di composizione.

54.11.5 Formato e aspetto delle pagine

L'etichetta *Page* consente di accedere alla definizione delle pagine, nel caso in cui la composizione richiesta serva a generare un formato PostScript o PDF. È possibile indicare il formato della pagina (corrispondente all'opzione '**`--size`** ***formato***'), se la stampa avviene su entrambi i lati del foglio (la voce DOUBLE-SIDED, ovvero l'opzione '**`--duplex`**'), i margini (le opzioni '**`--left`** ***n*****`{in|cm|mm}`**', '**`--right`** ***n*****`{in|cm|mm}`**', '**`--top`** ***n*****`{in|cm|mm}`**', '**`--bottom`** ***n*****`{in|cm|mm}`**'), l'intestazione e il piè pagina (opzioni '**`--header`** ***xyz***' e '**`--footer`** ***xyz***').

Figura 54.200. Definizione della pagina nel caso di composizione per la stampa.

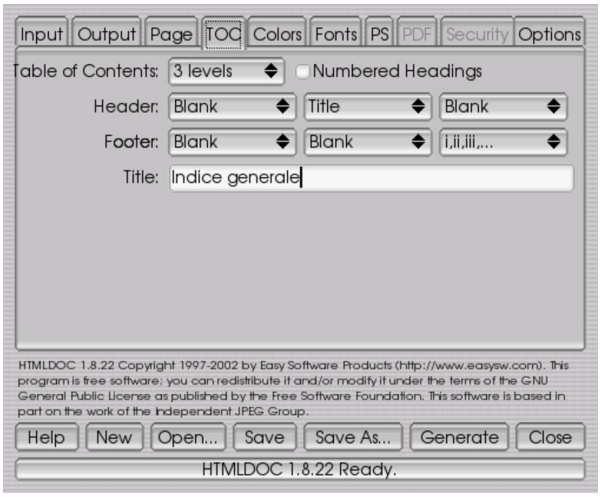


La selezione di intestazione e piè di pagina consente di definire ciò che deve apparire a sinistra, al centro e a destra di queste.

54.11.6 Indice generale

L'etichetta *TOC* consente di accedere alla definizione dell'indice generale, che in particolare prevede l'indicazione del livello di dettaglio che deve avere e consente la specificazione di intestazione e piè pagina differenti dal resto del documento.

Figura 54.201. Definizione dell'indice generale e della numerazione delle sezioni.

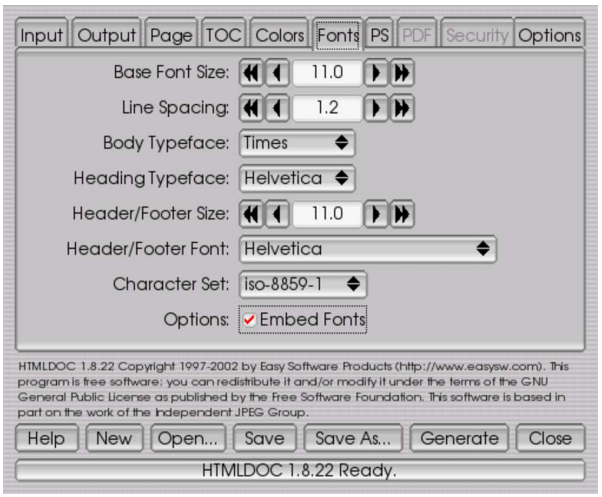


Il livello di dettaglio dell'indice si definisce anche con l'opzione '**`--toclevels` *n***', l'intestazione e il piè di pagina si possono definire con le opzioni '**`--tocheader` *xyz***' e '**`--tocfooter` *xyz***'. Infine, nella maschera si può vedere la voce NUMBERED HEADINGS, corrispondente all'opzione '**`--numbered`**', con cui si ottiene la numerazione delle sezioni (gli elementi '**`Hn`**').

54.11.7 Carattere da stampa

Sempre nel caso di composizione per la stampa, l'etichetta *Fonts* consente di definire il tipo e la dimensione dei caratteri da usare per il corpo, le sezioni, l'intestazione e il piè pagina. Per la necessità di essere compatibili al massimo, sono disponibili solo i tipi Times, Helvetica e Courier. Si osservi che l'altezza delle righe viene espressa in rapporto rispetto all'altezza dei caratteri, dove per esempio '**`1.2`**' rappresenta il 120 %.

Figura 54.202. Definizione del carattere da stampa.

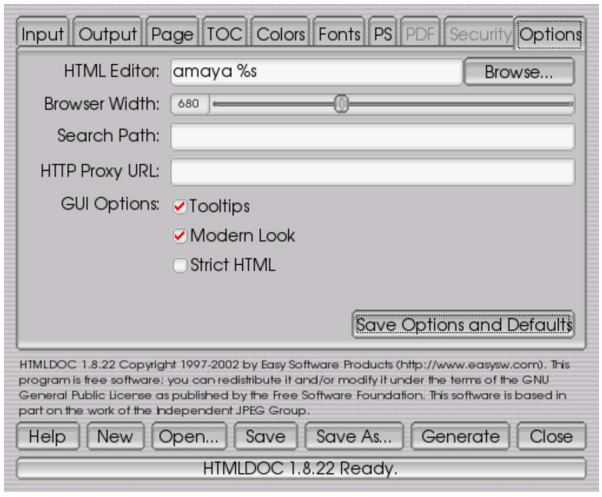


Le opzioni corrispondenti sono: '**`--bodyfont` *carattere***', '**`--fontsize` *n***' e '**`--fontspacing` *n***' per definire il carattere, la dimensione in punti e l'altezza delle righe nel testo normale; '**`--headfootfont` *carattere***' e '**`--headfootsize` *n***', per indicare il tipo di carattere e la dimensione in punti dell'intestazione e del piè di pagina; '**`--headingfont` *carattere***' per definire il tipo di carattere dei titoli delle sezioni.

54.11.8 Altre opzioni

L'ultima etichetta, *Options*, riguarda le opzioni rimanenti che non hanno trovato una collocazione migliore. In questo contesto viene definito in particolare il programma da utilizzare per la modifica dei file HTML del sorgente. Si può usare un programma per la modifica dei file di testo, oppure un applicativo più elaborato, specifico per i file HTML.

Figura 54.203. Definizione delle opzioni rimanenti.



54.11.9 Programmazione della composizione

HTMLDOC, quando funziona in modo interattivo, consente di conservare in un file l'impostazione, ovvero la configurazione di un documento, memorizzando tutte le opzioni selezionate. Quanto mostrato nelle figure di esempio, si tradurrebbe nel contenuto seguente:

```
#HTMLDOC 1.8.22
-t ps2 -f prova.ps --book --toclevels 3 --no-numbered
--toctitle "Indice generale" --title
--linkstyle underline --size A4 --left 3cm --right 3cm
--top 3cm --bottom 3cm --header .t. --footer h.1 --nup 1
--tocheader .t. --tocfooter ..i --portrait
--color --no-pscommands --no-xrxcomments --compression=1
--jpeg=0 --fontsize 11.0 --fontspacing 1.2
--headingfont Helvetica --bodyfont Times
--headfootsize 11.0
--headfootfont Helvetica --charset iso-8859-1
--browserwidth 680
uno.html due.html tre.html
```

Da *htmldoc(1)* si intuisce che l'autore suggerisca di usare l'estensione '`.book`' per questi file; tuttavia HTMLDOC non propone alcunché.

L'osservazione di questi file consente anche di capire meglio l'uso delle opzioni della riga di comando. In questo caso, volendo usare l'eseguibile '**`htmldoc`**' in modo non interattivo per fare la stessa cosa, il comando avrebbe potuto essere quello seguente:

```
$ htmldoc -t ps2 -f prova.ps --book --toclevels 3
        --no-numbered --toctitle "Indice generale"
        --title --linkstyle underline --size A4
        --left 3cm --right 3cm --top 3cm
        --bottom 3cm --header .t. --footer h.1
        --nup 1 --tocheader .t. --tocfooter ..i
        --portrait --color --no-pscommands
        --no-xrxcomments --compression=1 --jpeg=0
        --fontsize 11.0 --fontspacing 1.2
        --headingfont Helvetica --bodyfont Times
        --headfootsize 11.0 --headfootfont Helvetica
        --charset iso-8859-1 --browserwidth 680
        uno.html due.html tre.html[Invio]
```

Eccezionalmente, l'eseguibile '**`htmldoc`**' può ricevere come argomento il nome di uno di questi file '`.book`', ottenendo un funzionamento interattivo, partendo dalla configurazione contenuta nel file stesso. Supponendo di avere salvato quella configurazione nel file '`prova.book`', basterebbe riavviare nel modo seguente:

```
$ htmldoc prova.book [Invio]
```

Tabella 54.205. Riepilogo delle opzioni più comuni.

Opzione	Descrizione
`--webpage`	Il sorgente non è realizzato secondo la struttura di un libro.
`--format {ps|ps1|ps2|ps3|pdf|pdf11|pdf12|pdf13|pdf14|html}` `-t {ps|ps1|ps2|ps3|pdf|pdf11|pdf12|pdf13|pdf14|html}`	Formato finale della composizione.
`--outfile` ***file*** `-f` ***file***	File da generare dalla composizione.
`--outdir` ***directory*** `-d` ***directory***	Directory in cui inserire i file della composizione.
`--top` ***n*** `[pt|in|cm|mm]`	Margine superiore secondo l'unità di misura specificata. Se non si specifica l'unità di misura, si fa riferimento a punti tipografici ('**pt**').
`--bottom` ***n*** `[pt|in|cm|mm]`	Margine inferiore secondo l'unità di misura specificata. Se non si specifica l'unità di misura, si fa riferimento a punti tipografici ('**pt**').
`--left` ***n*** `[pt|in|cm|mm]`	Margine sinistro secondo l'unità di misura specificata. Se non si specifica l'unità di misura, si fa riferimento a punti tipografici ('**pt**').
`--right` ***n*** `[pt|in|cm|mm]`	Margine destro secondo l'unità di misura specificata. Se non si specifica l'unità di misura, si fa riferimento a punti tipografici ('**pt**').
`--size {letter|legal|a4|universal}`	Formato della carta in base a un nome.
`--size` ***larghezza*`x`*altezza* `[pt|in|cm|mm]`	Formato della carta espresso esattamente. Se non si specifica l'unità di misura, si fa riferimento a punti tipografici ('**pt**').
`--duplex` `--no-duplex`	Richiede che il formato generato sia o non sia predisposto per la stampa su entrambi i lati del foglio.
`--header` ***xyz*** `--footer` ***xyz***	Intestazione e piè di pagina.
`--tocheader` ***xyz*** `--tocfooter` ***xyz***	Intestazione e piè di pagina dell'indice generale.
`--headfootfont {courier|times|helvetica|symbol}`	Carattere da usare nelle intestazioni e nei piè di pagina.
`--headfootsize` ***n***	Dimensione del carattere da usare nelle intestazioni e nei piè di pagina.
`--headingfont {courier|times|helvetica|symbol}`	Carattere da usare nei titoli delle sezioni.

Opzione	Descrizione
`--bodyfont {courier|times|helvetica|symbol}` `--textfont {courier|times|helvetica|symbol}`	Carattere da usare nel corpo.
`--fontsize` ***n***	Dimensione del carattere normale in punti.
`--fontspacing` ***n***	Altezza della riga rispetto alla dimensione del carattere.
`--logoimage` ***file***	Definisce il file da usare come logo.
`--bodyimage` ***file***	Definisce il file da usare come sfondo di tutte le pagine.
`--titlefile` ***file*** `--titleimage` ***file***	File da usare per la copertina (può essere anche un'immagine).
`--book`	Specifica che il sorgente HTML è strutturato in modo appropriato alla composizione di un libro.
`--charset iso-8859-`***n***	Specifica la codifica da usare, secondo la codifica ISO 8859-***n***. Sono comunque disponibili anche altre codifiche espresse secondo parole chiave differenti.
`--no-toc`	Non genera l'indice generale.
`--toclevels` ***n***	Definisce il numero di livelli dell'indice generale.
`--toctitle "`***titolo***`"`	Definisce il titolo dell'indice generale.
`--title` `--no-title`	Genera o non genera la copertina.
`--embedfonts` `--no-embedfonts`	Richiede o meno l'inclusione di tutte le informazioni sui caratteri tipografici usati per la composizione PostScript o PDF.
`--links` `--no-links`	Genera o non genera i riferimenti ipertestuali nella composizione in formato PDF.
`--linkstyle {plain|underline}`	Dichiara lo stile con cui evidenziare i riferimenti ipertestuali.
`--numbered` `--no-numbered`	Numera o non numera le sezioni del documento.
`--nup {1|2|4|6|9|16}`	Mette assieme più pagine virtuali per ogni pagina fisica.

La tabella 54.205 riassume le funzionalità delle opzioni principali di HTMLDOC quando viene usato in modo non interattivo. È importante tenere in considerazione alcune cose, che vengono descritte brevemente nel seguito.

È possibile gestire solo immagini in formato GIF, JPG e PNG; in particolare, quando si ottiene una conversione in HTML, si usano le stesse immagini di partenza.

La dimensione della carta può essere indicata per nome, oppure direttamente, fornendo larghezza e altezza, nella forma ***m***`x`***n***, dove ***m*** rappresenta la larghezza e ***n*** l'altezza. Si osservi che questi due numeri vanno completati con l'indicazione finale dell'unità di misura (come si vede nella tabella), una volta sola per entrambi i valori. Il formato speciale denominato '**`universal`**' si riferisce a un compro-

messo tra il formato A4 e il formato lettera (8,5 in × 11 in), in modo che possa essere stampato con entrambi i tipi di carta; in pratica si utilizza la larghezza del formato A4 e l'altezza del formato lettera.

Le opzioni che definiscono il contenuto delle intestazioni e dei piè di pagina, utilizzano una simbologia speciale, in cui tre lettere indicano rispettivamente la parte sinistra, quella centrale e quella destra della riga. La tabella 54.206 riepiloga questi simboli.

Tabella 54.206. Simboli usati negli argomenti delle opzioni che definiscono il contenuto delle intestazioni e dei piè di pagina.

Simbolo	Descrizione
`.`	Vuoto.
`:`	Numero della pagina del capitolo in relazione alla quantità totale di pagine del capitolo, secondo la forma: '***n_pagina*/*n_complessivo***'.
`/`	Numero della pagina del documento in relazione alla quantità totale di pagine del documento, secondo la forma: '***n_pagina*/*n_complessivo***'.
`t`	Titolo del documento.
`c`	Capitolo attuale.
`h`	Sezione attuale.
`d`	Data.
`T`	Orario.
`D`	Data e orario.
`l`	Logo.
`1`	Pagina del documento in numero arabo normale.
`a`	Pagina del documento in lettere minuscole.
`A`	Pagina del documento in lettere maiuscole.
`i`	Pagina del documento in numero romano minuscolo.
`I`	Pagina del documento in numero romano maiuscolo.
`C`	Pagina del capitolo in numero arabo normale.

54.11.10 Informazioni particolari nel sorgente

«

All'interno del sorgente HTML, alcune informazioni vengono trattate in modo speciale da HTMLDOC. Si tratta in special modo di commenti SGML che acquistano il significato di direttive, che spesso vanno a sovrapporsi a opzioni della riga di comando di HTMLDOC. La tabella 54.207 successiva descrive alcune di queste direttive.

Tabella 54.207. Alcune direttive in forma di commenti SGML.

Direttiva	Descrizione
`<-- MEDIA SIZE "`***dimensione***`" -->`	Dimensioni della pagina. Può trattarsi delle parole chiave '**`letter`**', '**`legal`**', '**`universal`**', '**`A4`**', oppure delle dimensioni espresse secondo la forma '***m*x*n*[pt|in|cm|mm]**'.

`<-- MEDIA TOP `***n***`[pt|in|cm|mm] -->`
`<-- MEDIA BOTTOM `***n***`[pt|in|cm|mm] -->`
`<-- MEDIA LEFT `***n***`[pt|in|cm|mm] -->`
`<-- MEDIA RIGHT `***n***`[pt|in|cm|mm] -->`
	Margine: superiore, inferiore, sinistro, destro.

`<-- MEDIA LANDSCAPE YES|NO -->`	Abilita o disabilita la composizione orizzontale.
`<-- MEDIA DUPLEX YES|NO -->`	Abilita o disabilita la composizione adatta per la stampa fronte-retro.

Direttiva	Descrizione

`<-- HEADER LEFT "`***testo***`" -->`
`<-- HEADER CENTER "`***testo***`" -->`
`<-- HEADER RIGHT "`***testo***`" -->`
	Parte sinistra, centrale e destra del testo da mettere come intestazione della pagina attuale.

`<-- FOOTER LEFT "`***testo***`" -->`
`<-- FOOTER CENTER "`***testo***`" -->`
`<-- FOOTER RIGHT "`***testo***`" -->`
	Parte sinistra, centrale e destra del testo da mettere alla base della pagina attuale.

`<-- HALF PAGE -->`	Salta alla prossima mezza pagina.
`<-- NEW PAGE -->` `<-- BREAK PAGE -->`	Salta all'inizio della prossima pagina.
`<-- NEED `***n_righe***` -->` `<-- NEED `***n***`{pt|in|cm|mm} -->`	Salta all'inizio della prossima pagina se non è disponibile lo spazio verticale richiesto.
`<-- NUMBER-UP {1|2|4|6|9|16} -->`	Mette assieme più pagine virtuali per ogni pagina fisica.

Per comporre il testo delle intestazioni e dei piè di pagina, si possono usare dei «simboli» speciali, che assomigliano a variabili. Si tratta precisamente di nomi prefissati da un dollaro ('**$**'), che vengono espansi in fase di composizione. La tabella 54.208 descrive alcuni di questi simboli.

Tabella 54.208. Alcuni simboli usati nelle intestazioni e nei piè di pagina, dichiarati attraverso le direttive speciali in forma di commenti SGML.

Simbolo	Descrizione
`$$`	Rappresenta un dollaro singolo.
`$TITLE`	Inserisce il titolo del documento.
`$CHAPTER`	Inserisce il titolo della sezione corrispondente al capitolo corrente.
`$HEADING`	Inserisce il titolo della sezione corrente.
`$CHAPTERPAGE` `$CHAPTERPAGE(1|i|I|a|A)`	Inserisce il numero di pagina del capitolo, eventualmente secondo il formato stabilito dalla lettera tra parentesi: '**`1`**' numero arabo normale; '**`i`**' numero romano minuscolo; '**`I`**' numero romano maiuscolo; '**`a`**' sequenza alfabetica minuscola; '**`A`**' sequenza alfabetica maiuscola.
`$CHAPTERPAGES` `$CHAPTERPAGES(1|i|I|a|A)`	Inserisce il numero complessivo di pagine del capitolo, eventualmente secondo il formato stabilito dalla lettera tra parentesi: '**`1`**' numero arabo normale; '**`i`**' numero romano minuscolo; '**`I`**' numero romano maiuscolo; '**`a`**' sequenza alfabetica minuscola; '**`A`**' sequenza alfabetica maiuscola.
`$PAGE` `$PAGE(1|i|I|a|A)`	Inserisce il numero di pagina assoluto, eventualmente secondo il formato stabilito dalla lettera tra parentesi: '**`1`**' numero arabo normale; '**`i`**' numero romano minuscolo; '**`I`**' numero romano maiuscolo; '**`a`**' sequenza alfabetica minuscola; '**`A`**' sequenza alfabetica maiuscola.

Simbolo	Descrizione
`$PAGES`	Inserisce il numero complessivo di pagine assoluto del documento, eventualmente secondo il formato stabilito dalla lettera tra parentesi: '**1**' numero arabo normale; '**i**' numero romano minuscolo; '**I**' numero romano maiuscolo; '**a**' sequenza alfabetica minuscola; '**A**' sequenza alfabetica maiuscola.
`$DATE`	Inserisce la data.
`$TIME`	Inserisce l'ora.
`$LOGOIMAGE`	Inserisce soltanto il logo, ignorando il resto del testo.

HTMLDOC trae delle informazioni da alcuni elementi '**META**' comuni. La tabella 54.209 descrive alcuni tipi riconosciuti e il significato loro attribuito.

Tabella 54.209. Elementi '**META**' riconosciuti da HTMLDOC.

Sintassi	Descrizione
`<META NAME="AUTHOR" CONTENT="`***autore***`">`	Autore del documento.
`<META NAME="COPYRIGHT" CONTENT="`***copyright***`">`	Copyright del documento.
`<META NAME="GENERATOR" CONTENT="`***applicazione***`">`	Applicazione che ha generato il sorgente HTML.
`<META NAME="KEYWORDS" CONTENT="`***parole_chiave***`">`	Elenco delle parole chiave più importanti, che riconducono ai contenuti del documento.
`<META NAME="SUBJECT" CONTENT="`***genere***`">`	Genere a cui appartiene il documento.

54.12 Motori di ricerca e robot

Più passa il tempo e più sono i documenti che vengono pubblicati su Internet. I motori di ricerca, ovvero i servizi che gestiscono gli indici delle pubblicazioni, sono sempre più sommersi di lavoro. In questa situazione, ognuno applica una propria politica di filtro dei documenti che vengono sottoposti per l'inclusione nel loro indice. In generale, non basta realizzare un documento HTML corretto, occorre pensare anche ai motori di ricerca.

Il documento HTML, per poter essere preso in considerazione in modo corretto dai motori di ricerca, deve avere una serie di elementi '**META**' nell'intestazione, contenenti alcune informazioni salienti. Ciò permette la classificazione del documento e la creazione di indici chiari per l'utente di quel servizio. Tuttavia, il problema è che non tutti i motori di ricerca utilizzano le stesse informazioni nello stesso modo; così, ci si affida generalmente all'esperienza degli altri per la compilazione di tali elementi. Qui si raccolgono solo alcune indicazioni che, però, potrebbero essere superate facilmente con il passare del tempo.

54.12.1 Elementi META

Gli elementi '**META**' sono vuoti, nel senso che non delimitano alcun testo, e si collocano nell'intestazione del file HTML, ovvero nell'elemento '**HEAD**'. Nella maggior parte dei casi, l'elemento '**META**' si utilizza con l'attributo '**name**' e l'attributo '**content**', attraverso i quali si stabilisce un nome a cui viene assegnato un contenuto.

Il DTD dell'HTML non stabilisce quali siano i nomi che si possono usare per l'attributo '**name**' e da questo nascono tutti i problemi. In particolare, c'è da considerare che alle volte i nomi e i valori abbinati non fanno differenza tra maiuscole e minuscole, altre volte pare che la facciano.

L'esempio seguente mostra un caso tipico di utilizzo per un documento realizzato in italiano:


```
<!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
<HTML LANG="it">
<HEAD>
    <META HTTP-EQUIV="Content-Type"
          CONTENT="text/html; charset=UTF-8">
    <META NAME="generator" CONTENT="ALtools">
    <META NAME="description"
          CONTENT="GNU/Linux e il software libero"
          LANG="it">
    <META NAME="description"
          CONTENT="GNU/Linux and free software" LANG="en">
    <META NAME="keywords"
          CONTENT="GNU/Linux, Unix, software, software
                   libero, free software">
    <META NAME="distribution" CONTENT="Global" LANG="en">
    <META NAME="rating" CONTENT="General" LANG="en">
    <META NAME="resource-type" CONTENT="document" LANG="en">
    <META NAME="classification" CONTENT="computers"
          LANG="en">
    <META NAME="revisit-after" CONTENT="7 days"
          LANG="en">
    <META NAME="ROBOTS" CONTENT="ALL">
    <META NAME="SPIDERS" CONTENT="ALL">
    <META NAME="author" CONTENT="Daniele Giacomini">
    <META NAME="copyright"
          CONTENT="© 1997-2001 Daniele Giacomini">
    <TITLE>Appunti Linux</TITLE>
    ...
</HEAD>
...
```

Il significato di queste informazioni dovrebbe essere intuitivo, salvo qualche caso, ma in particolare è necessario osservare un problema: alcune cose sono espresse attraverso sigle o parole chiave che hanno significato per la lingua inglese, mentre potrebbero essere attese parole o definizioni diverse nel caso di un documento in italiano. Nell'esempio si può osservare che l'elemento '**HTML**' possiede l'attributo '**lang**' a cui è assegnato il valore '**it**', allo scopo di indicare che tutto il documento è scritto in lingua italiana. Pertanto, per modificare questo assunto negli elementi '**META**' in cui il linguaggio può avere importanza, è stato aggiunto nuovamente l'attributo '**lang**' con il valore '**en**'. Può darsi che questa precauzione non serva a nulla, ma potrebbe essere importante in futuro.

Eventualmente, si potrebbe anche arrivare a duplicare alcune informazioni per diversi linguaggi. Per esempio, l'informazione denominata '**description**' viene fornita due volte: prima in italiano e poi in inglese.

L'elenco seguente descrive brevemente le informazioni più importanti che si possono dare in questo modo.

Valore assegnato	Descrizione
`description` `Description`	Si tratta di una descrizione breve del contenuto che potrebbe essere mostrato negli indici. A titolo indicativo, non dovrebbe superare le 25 parole, per essere certi che sia presa in considerazione integralmente.
`keywords` `Keywords`	Si tratta di un elenco di parole, o frasi brevi, separate da una virgola. Queste parole rappresentano gli argomenti principali del documento. Indicandole in questo modo, si cerca di farle risaltare (anche se nel documento vengono usate poco o non vengono usate affatto), per far sì che vengano prese in considerazione in modo particolare. A titolo indicativo, l'elenco non dovrebbe superare le 25 parole, per essere certi che questo venga preso in considerazione. Si intuisce che le prime parole di questo elenco devono essere le più importanti.
`distribution` `Distribution`	Probabilmente si riferisce all'estensione che ha o può avere la diffusione del documento. Le parole che possono essere assegnate sono '**Global**' e '**Local**', con i significati che si possono intuire.

Valore assegnato	Descrizione
`rating` `Rating`	Probabilmente si riferisce al tipo di pubblico a cui si rivolge il documento. In generale viene assegnata solo la parola chiave '**`General`**'; qualcuno suggerisce anche l'uso di, '**`Mature`**' e '**`Restricted`**', ma il significato in pratica non è chiaro.
`classification` `Classification`	Si tratta della classificazione del contenuto del documento. È difficile fare un elenco dei termini che si possono usare, perché dipendono dal motore di ricerca. Probabilmente si può trattare di: '**`business`**', '**`computers`**', '**`entertainment`**', '**`internet`**', '**`miscellaneous`**', '**`personal`**'.
`resource-type`	Si tratta della definizione che si dà al documento HTML. Da quanto si vede, si usa sempre solo la parola chiave '**`document`**' (solo in minuscolo).
`revisit-after`	Apparentemente, questa indicazione serve a richiedere al motore di ricerca di ripassare dopo un certo numero di giorni. Non è garantito il successo di questa richiesta, ma nulla vieta di provarci.
`ROBOTS`	Questa informazione serve a chiedere esplicitamente o a vietare la scansione e l'indicizzazione. In generale si assegna la parola chiave '**`ALL`**' perché venga preso in considerazione il documento a tutti gli effetti, assieme ai riferimenti a cui punta, mentre si usa la parola chiave '**`INDEX`**' per richiedere la sola indicizzazione e '**`FOLLOW`**' per seguire i riferimenti. Per evitare l'indicizzazione si usa '**`NOINDEX`**', mentre per evitare di seguire i riferimenti si usa '**`NOFOLLOW`**'. Qualcuno suggerisce di utilizzare la stringa '**`ALL,INDEX,FOLLOW`**' per ottenere il risultato migliore.
`SPIDERS`	Apparentemente funziona nello stesso modo di '**`ROBOTS`**' e probabilmente accetta gli stessi valori.

54.12.2 Filtro iniziale alla scansione dei robot

Nel momento in cui si è posto il problema dell'esistenza di tutta una serie di servizi di scansione della documentazione su Internet, si è pensato all'opportunità di bloccare, in certe circostanze, il lavoro di questi «robot». Gli amministratori dei servizi HTTP hanno la possibilità di realizzare il file '`/robots.txt`', contenente l'indicazione dei percorsi che non devono essere scanditi.

Anche se si tratta di un compito che riguarda gli amministratori, è opportuno sapere leggere le istruzioni di questo file, nel caso esista, per sapere se il proprio documento può essere raggiunto o meno dai motori di ricerca e da altri servizi simili.

Il file in questione, collocato all'inizio della gerarchia del servizio HTTP a cui si riferisce, è un file di testo normale, in cui si indicano dei commenti, preceduti dal simbolo '`#`', e una serie di campi nella forma:

```
campo: valore
```

Le informazioni di questo file sono suddivise in base al nome del programma robot che si vuole filtrare:

```
User-agent: nome
```

Uno o più campi del genere, posti di seguito, iniziano la definizione del filtro riferito ai programmi rispettivi. Se al posto del nome si indica un asterisco, si intendono simultaneamente tutti i programmi che non siano stati presi in considerazione diversamente.


```
Disallow: [percorso]
```

Il campo '**`Disallow`**' serve a specificare un percorso da escludere dalla scansione dei robot presi in considerazione.[8]

```
# http://www.brot.dg/robots.txt
User-agent: *
Disallow /tmp/
Disallow /cgi-bin/
Disallow /prova.html
```

Supponendo che l'esempio si riferisca al file '`http://www.brot.dg/robots.txt`', si mostra il caso in cui si vogliono escludere tutti i robot dal contenuto di '`http://www.brot.dg/tmp/`', '`http://www.brot.dg/cgi-bin/`' e dal file '`http://www.brot.dg/prova.html`'.

```
# http://www.brot.dg/robots.txt
User-agent: *
Disallow
```

In questo caso non si esclude alcunché.

```
# http://www.brot.dg/robots.txt
User-agent: *
Disallow /
```

Questo nuovo esempio esclude l'accesso a tutto il servizio.

54.13 Riferimenti

- T. Berners-Lee, R. Fielding, U.C. Irvine, L. Masinter, *RFC 2396: Uniform Resource Identifiers (URI): General Syntax*, 1998, http://www.ietf.org/rfc/rfc2396.txt
- Wikipedia, *Uniform resource name*, http://en.wikipedia.org/wiki/Uniform_resource_name
- *Using DNS as a URN resolver*, http://stackoverflow.com/questions/5476121/using-dns-as-a-urn-resolver
- International ISBN agency, *The ISBN Users' Manual*, http://www.isbn.org/standards/home/isbn/International/ISBNmanual.asp
- W3C, *MarkUp Validation Service*, http://validator.w3.org/
- W3C, *Link Checker*, http://validator.w3.org/checklink
- *ISO/IEC 15445:2000*, http://www.scss.tcd.ie/misc/15445/15445.html , https://www.scss.tcd.ie/misc/15445/15445.HTML
- Wikipedia, *HTML5*, http://it.wikipedia.org/wiki/HTML5
- *Character sets*, http://www.iana.org/assignments/character-sets
- W3C, *Technical Reports and Publications*, http://www.w3.org/TR/
  - *HTML 4.01 Specification*, http://www.w3.org/TR/html401/
  - *XHTML 1.0: The Extensible HyperText Markup Language*, http://www.w3.org/TR/xhtml1/
  - *Cascading Style Sheets*, http://www.w3.org/TR/REC-CSS1 , http://www.w3.org/TR/CSS2/
- FunctionX, *JavaScript tutorial*, http://www.functionx.com/javascript/
- Netscape, *Client-Side JavaScript Guide*, http://docs.oracle.com/cd/E19957-01/816-6408-10/contents.htm
- Michele Sciabarrà, *Linux e programmazione web*, McGraw-Hill, 2001, ISBN 88-386-4177-3 (in particolare il capitolo *JavaScript*)
- Jan Kärrman, *Using html2ps*
- Irène Vatton, *Amaya documentation*, http://www.w3.org/Amaya/User/
- *Amaya user manual*, http://www.w3.org/Amaya/User/doc/Manual.html
- *Amaya Home Page*, http://www.w3.org/Amaya/ , http://www.w3.org/Amaya/User/BinDist.html

• *TheFreeSite.com*, *http://www.thefreesite.com*

[1] Qui viene usato il termine «directory», ma in pratica potrebbe anche non essere esattamente una directory vera e propria.

[2] Generalmente si preferisce il formato JPG, perché originariamente il formato PNG era accettato solo da alcuni programmi di navigazione, mentre il formato GIF era brevettato.

[3] Per motivi di compatibilità con i vecchi navigatori, i marcatori di questo genere vanno indicati avendo l'accortezza di lasciare uno spazio prima della barra finale; per esempio: '`<br />`'.

[4] In generale, il buon senso dovrebbe essere sufficiente per intendere quando una caratteristica viene ereditata e quando questo non può succedere.

[5] **HTML2ps** GNU GPL

[6] **Amaya** software libero con licenza speciale

[7] **HTMLDOC** GNU GPL

[8] Non è possibile indicare dei metacaratteri (caratteri jolly) nel percorso; d'altra parte ciò non avrebbe significato, dal momento che si intendono tutti i percorsi che iniziano come indicato e proseguono poi in qualunque modo.

HTML: lezioni pratiche e verifiche

Prima di iniziare le lezioni pratiche sulla realizzazione di documenti HTML, è bene predisporre o prepararsi a utilizzare degli strumenti adatti. Le pagine vanno scritte utilizzando un programma per la creazione e modifica di file di testo, come può essere VI, Mcedit, Gedit o qualunque altro programma appropriato, di cui si conosca l'utilizzo.

I file vanno scritti utilizzando la codifica ASCII pura e semplice, pertanto, non possono essere usate lettere accentate se non utilizzando le macro appropriate, come viene descritto durante le esercitazioni.

Per poter controllare l'aspetto finale del file HTML che si sta scrivendo con un programma di scrittura per file di testo, occorre usare un navigatore (come Mozilla, Amaya o altro). Se possibile, è utile confrontare l'aspetto finale del proprio lavoro con navigatori differenti.

Le verifiche richiedono la stampa del risultato finale, attraverso un navigatore, assieme alla stampa del sorgente.

Prima di stampare il proprio lavoro è necessario accertarsi della correttezza sintattica di ogni file HTML. Per questo possono essere utilizzati diversi strumenti, a seconda di ciò che è disponibile presso il proprio elaboratore. Se è stato installato correttamente il pacchetto SP (*SGML parser*) con i DTD dei vari formati HTML, è sufficiente il comando seguente:

```
$ nsgmls -s file_html 2>&1 | less [Invio]
```

In mancanza di altro, si può accedere al servizio offerto da consorzio W3C presso l'indirizzo *http://validator.w3.org/file-upload.html* . Si osservi che è necessario indicare la codifica e può essere scelta la sigla '**iso-8859-1**'.

Si osservi che per la valutazione delle verifiche si tiene conto anche della coerenza estetica del sorgente, dove, in particolare, non vanno superate le 80 colonne di larghezza del testo.

All'interno di queste lezioni sono richieste alcune verifiche, che vengono riepilogate qui, per facilitare la ripresa del lavoro quando questo viene sospeso.

1. Verifica sulla struttura generale e sull'uso delle macro, sezione 55.2.3.
2. Verifica sull'uso degli elementi interni al testo lineare, sezioni 55.3.3 e 55.3.4.
3. Verifica sull'uso degli elementi per il testo preformattato, sezione 55.4.4.
4. Verifica sull'uso degli elenchi, sezione 55.5.2.
5. Verifica sull'uso di tabelle elementari, sezione 55.6.3.
6. Verifica sull'uso di tabelle complesse, sezione 55.7.2.
7. Verifica sull'uso dei riferimenti ipertestuali, sezione 55.8.2.
8. Verifica sull'inserzione di oggetti, sezione 55.10.3.
9. Verifica sull'inserzione di immagini, sezione 55.11.3.
10. Verifica sull'uso delle proprietà riferite ai caratteri, sezione 55.14.3.
11. Verifica sull'uso delle proprietà riferite ai colori, sezione 55.15.5.
12. Verifica sull'uso delle proprietà riferite al testo, sezione 55.16.2.
13. Verifica sull'uso delle proprietà riferite ai blocchi, sezione 55.17.4.
14. Verifica conclusiva, sezione 55.19.1.

55.1 Struttura generale e intestazione

Un documento HTML si compone di un file di testo, all'interno del quale le informazioni sono circoscritte da ***elementi***, delimitati da ***marcatori*** che aprono e chiudono gli elementi stessi. Per esempio,

il marcatore '**<HTML>**' apre l'elemento '**HTML**', mentre il marcatore '**</HTML>**' lo chiude.

Il file di testo, che costituisce il ***sorgente*** del documento o della pagina HTML, può contenere dei commenti, che non vengono mostrati dal navigatore. Questi commenti o annotazioni si delimitano tra '**<!--**' e '**-->**'.

55.1.1 File «struttura-01.html»

Si scriva il testo seguente salvandolo nel file '`struttura-01.html`':

```
1  <!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
2  <HTML LANG="it">
3  <HEAD>
4      <META HTTP-EQUIV="Content-Type"
5            CONTENT="text/html; charset=us-ascii">
6      <META NAME="Description"
7            CONTENT="Struttura HTML, esempio 01">
8      <META NAME="Keywords" CONTENT="HTML, esempio">
9      <META NAME="Author"
10           CONTENT="Tizio Tizi, tizio@brot.dg">
11     <META NAME="Date" CONTENT="2012.01.01">
12     <META NAME="Resource-type" LANG="en"
13           CONTENT="Document">
14     <META NAME="Revisit-after" LANG="en"
15           CONTENT="15 days">
16     <META NAME="Robots" CONTENT="ALL">
17     <TITLE>Struttura di un documento HTML</TITLE>
18 </HEAD>
19 <BODY>
20
21 </P>Struttura di un documento HTML. Bla bla bla bla bla
22 bla bla bla bla bla bla bla bla bla bla bla bla bla bla
23 bla bla bla bla bla bla bla bla bla bla bla bla bla bla
24 bla bla bla bla bla bla bla bla bla bla bla bla.<P>
25
26 </BODY>
27 </HTML>
```

Questo primo esempio contiene un errore; lo si scopre facilmente con l'ausilio di uno strumento di controllo, come descritto nella premessa del capitolo:

```
$ nsgmls -s struttura-01.html 2>&1 | less [Invio]

 nsgmls:struttura-01.html:21:3:E: end tag for element "P"
 which is not open
 nsgmls:struttura-01.html:21:4:E: character data is not
 allowed here
```

Dall'analisi si viene a sapere che nella riga numero 21 e alla colonna numero 3 del file '`struttura-01.html`' appare un marcatore di chiusura per un elemento '**P**' che non è stato aperto; di conseguenza, alla riga 21, colonna 4, inizia del testo che non è ammissibile.

Osservando il sorgente proposto si comprende che il blocco di testo che inizia dalla riga numero 21 andrebbe scritto nel modo seguente:

```
21 <P>Struttura di un documento HTML. Bla bla bla bla bla
22 bla bla bla bla bla bla bla bla bla bla bla bla bla bla
23 bla bla bla bla bla bla bla bla bla bla bla bla bla bla
24 bla bla bla bla bla bla bla bla bla bla bla bla.</P>
```

Si corregga l'errore, si salvi con lo stesso nome e si verifichi nuovamente la correttezza del file (se ci sono altri errori si provveda alla loro correzione, controllando nuovamente fino a che il file risulta corretto).

Un file HTML, scritto secondo lo standard ISO 15445, inizia con la dichiarazione che appare nella prima riga dell'esempio già mostrato:

```
1  <!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
```

Questa indicazione è obbligatoria e viene usata in tutti i file di queste lezioni pratiche. Dopo la dichiarazione del tipo di documento si osservi che c'è un solo elemento, denominato '**HTML**' (dalla riga numero 2 alla riga 22). Questo elemento contiene l'elemento '**HEAD**' (dalla riga numero 3 alla riga 13) e l'elemento '**BODY**' (dalla riga numero 14 alla riga 21).

L'elemento '**HEAD**' viene usato per contenere informazioni relative al documento che non vengono visualizzate dal navigatore, mentre l'elemento '**BODY**' contiene ciò che viene visto normalmente.

L'elemento '**HTML**' (si osservi il marcatore di apertura alla riga 2) contiene l'attributo '**LANG**', con il quale si dichiara l'utilizzo della lingua italiana per tutto il file, salvo indicazione diversa di elementi interni.

Gli elementi '**META**' contenuti nell'elemento '**HEAD**' sono descritti brevemente nella tabella successiva. Si osservi in particolare il fatto che sono tutti vuoti, pertanto non è stato scritto il marcatore di chiusura (essendo perfettamente superfluo).

Marcatore	Descrizione
`<META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=us-ascii">`	Dichiara il documento secondo lo standard MIME (*Multipurpose internet mail extensions*) e la codifica.
`<META NAME="Description" CONTENT="Struttura HTML, esempio 01">`	Dà una descrizione al documento, utile per i sistemi di indicizzazione delle informazioni che appaiono nella rete.
`<META NAME="Keywords" CONTENT="HTML, esempio">`	Dichiara le parole chiave più importanti per fare riferimento al documento attraverso i motori di ricerca.
`<META NAME="Author" CONTENT="Tizio Tizi, tizio@brot.dg">`	Dichiara le informazioni riferite agli autori del documento.
`<META NAME="Date" CONTENT="2012.01.01">`	Dichiara la data del documento.
`<META NAME="Resource-type" LANG="en" CONTENT="Document">`	Definisce il tipo di «risorsa», attraverso una denominazione che serve ai sistemi di indicizzazione delle informazioni.
`<META NAME="Revisit-after" LANG="en" CONTENT="15 days">`	Dice ai motori di ricerca di controllare nuovamente il file a intervalli di 15 giorni.
`<META NAME="Robots" CONTENT="ALL">`	Dichiara che è consentito espressamente a tutti i sistemi automatici di scansione dei documenti, come i motori di ricerca o i sistemi di indicizzazione, di acquisire tutto il contenuto.

Gli elementi '**META**' contengono principalmente informazioni che servono ai motori di ricerca, pertanto, alcune informazioni sono scritte espressamente in inglese, dichiarando il linguaggio con l'attributo '**LANG**', come eccezione rispetto alla dichiarazione complessiva contenuta nell'elemento '**HTML**'.

Alla fine degli elementi '**META**', alla riga numero 17, appare il titolo del documento, racchiuso nell'elemento '**TITLE**'.

Si salvi con il nome '`struttura-02.html`' questo file, modificando le informazioni dell'intestazione con qualcosa di appropriato, soprattutto mettendo al posto dell'autore i propri dati e indicando la data corretta del lavoro. Si ricordi di controllare la correttezza sintattica con gli strumenti già noti.

55.1.2 File «struttura-02.html»

Viene mostrato come dovrebbe apparire il file '`struttura-02.html`' dopo le modifiche richieste. Si osservi che il nome «Tizio Tizi», l'indirizzo di posta elettronica relativo e la classe devono essere sostituiti con i propri dati, eventualmente con un indirizzo di posta elettronica fasullo, se si preferisce mantenerlo anonimo:

```
1  <!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
2  <HTML LANG="it">
3  <HEAD>
4      <META HTTP-EQUIV="Content-Type"
5            CONTENT="text/html; charset=us-ascii">
6      <META NAME="Description"
7            CONTENT="Struttura HTML, esempio 02">
8      <META NAME="Keywords" CONTENT="HTML, esempio">
9      <META NAME="Author"
10           CONTENT="Tizio Tizi, tizio@brot.dg, classe 4Z">
11     <META NAME="Date" CONTENT="2012.01.01">
12     <META NAME="Resource-type" LANG="en"
13           CONTENT="Document">
14     <META NAME="Revisit-after" LANG="en"
15           CONTENT="15 days">
16     <META NAME="Robots" CONTENT="ALL">
17     <TITLE>Struttura di un documento HTML</TITLE>
18 </HEAD>
19 <BODY>
20
21 <P>Struttura di un documento HTML. Bla bla bla bla
22 bla bla bla bla bla bla bla bla bla bla bla bla bla bla
23 bla bla bla bla bla bla bla bla bla bla bla bla bla bla
24 bla bla bla bla bla bla bla bla bla bla bla bla.</P>
25
26 </BODY>
27 </HTML>
```

L'elemento '**`BODY`**' (righe da 19 a 26) contiene soltanto un elemento '**`P`**', che serve a delimitare un blocco di testo (la lettera «p» sta per «paragrafo»). Se si osserva il risultato che si ottiene da un navigatore, si può notare che il titolo non fa parte del documento visualizzato, ma appare probabilmente in una posizione esterna a questo.

Figura 55.7. Aspetto del file '`struttura-02.html`' con un navigatore.

Struttura di un documento HTML. 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 bla bla bla bla bla bla.

Dalla figura si può vedere che l'impaginazione del blocco di testo delimitato dall'elemento '**`P`**' dipende dalla dimensione del carattere e dall'ampiezza disponibile effettivamente, durante l'utilizzo del navigatore.

55.1.3 File «struttura-03.html»

Si salvi con il nome '`struttura-03.html`' il file usato in precedenza e lo si modifichi secondo il modello dell'esempio seguente:

```
<!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
<HTML LANG="it">
<HEAD>
    <META HTTP-EQUIV="Content-Type"
          CONTENT="text/html; charset=us-ascii">
    <META NAME="Description"
          CONTENT="Struttura HTML, esempio 03">
    <META NAME="Keywords" CONTENT="HTML, esempio">
    <META NAME="Author"
          CONTENT="Tizio Tizi, tizio@brot.dg, classe 4Z">
    <META NAME="Date" CONTENT="2012.01.01">
    <META NAME="Resource-type" LANG="en"
          CONTENT="Document">
    <META NAME="Revisit-after" LANG="en"
          CONTENT="15 days">
    <META NAME="Robots" CONTENT="ALL">
    <TITLE>Struttura di un documento HTML</TITLE>
</HEAD>
<BODY>

<!-- Capitolo -->
<H1>Struttura di un documento HTML</H1>

<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 bla bla bla bla
bla bla bla bla.</P>

<!-- Sezione -->
<H2>Intestazione</H2>

<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 bla bla bla bla
bla bla bla bla.</P>

<!-- Sezione -->
<H2>Corpo</H2>

<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 bla bla bla bla
bla bla bla bla.</P>

</BODY>
</HTML>
```

Gli elementi '**`H1`**' servono a delimitare il titolo di una sezione importante (di primo livello) del documento; gli elementi '**`H2`**' si usano per il titolo di una sezione di livello inferiore, continuando così a decrescere fino a '**`H6`**'. Si osservi come potrebbe risultare il documento di esempio nella figura successiva.

Figura 55.9. Aspetto del file '`struttura-03.html`' con un navigatore.

Struttura di un documento HTML

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 bla bla bla bla bla bla.

Intestazione

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 bla bla bla bla bla bla.

Corpo

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 bla bla bla bla bla bla.

L'esempio costituito dal file '`struttura-03.html`' si compone di un titolo di una sezione di livello uno, che contiene una descrizione e una serie di sezioni di livello due. Il significato che può avere una sezione di livello uno dipende dal contesto: se si trattasse di un libro, l'elemento '**`H1`**' potrebbe essere usato per i titoli dei capitoli.

55.1.4 File «struttura-04.html»

Si salvi con il nome '`struttura-04.html`' il file usato in precedenza e lo si modifichi secondo il modello dell'esempio seguente:

```
<!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
<HTML LANG="it">
<HEAD>
    <META HTTP-EQUIV="Content-Type"
          CONTENT="text/html; charset=us-ascii">
    <META NAME="Description"
          CONTENT="Struttura HTML, esempio 04">
    <META NAME="Keywords" CONTENT="HTML, esempio">
    <META NAME="Author"
          CONTENT="Tizio Tizi, tizio@brot.dg, classe 4Z">
    <META NAME="Date" CONTENT="2012.01.01">
    <META NAME="Resource-type" LANG="en"
          CONTENT="Document">
    <META NAME="Revisit-after" LANG="en"
          CONTENT="15 days">
    <META NAME="Robots" CONTENT="ALL">
    <TITLE>Struttura di un documento HTML</TITLE>
</HEAD>
<BODY>

<H1>Introduzione</H1>

<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 bla bla bla bla
bla bla bla bla.</P>

<H1>Struttura di un documento HTML</H1>

<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 bla bla bla bla
bla bla bla bla.</P>
```

```
<H2>Intestazione</H2>

<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 bla bla bla bla
bla bla bla bla.</P>

<H2>Corpo</H2>

<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 bla bla bla bla
bla bla bla bla.</P>

<H1>Appendice</H1>

<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 bla bla bla bla
bla bla bla bla.</P>

<H1>Indice analitico</H1>

<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 bla bla bla bla
bla bla bla bla.</P>

</BODY>
</HTML>
```

L'esempio dovrebbe far comprendere meglio l'utilizzo degli elementi '`H`*n*'. In questo caso il documento è strutturato come se si trattasse di un libro, composto da un'introduzione, un capitolo, un'appendice e un indice analitico.

Figura 55.11. Aspetto del file '`struttura-04.html`' con un navigatore.

Introduzione

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 bla bla bla bla bla bla bla bla bla bla.

Struttura di un documento HTML

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 bla bla bla bla bla bla bla bla bla bla.

Intestazione

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 bla bla bla bla bla bla bla bla bla bla.

Corpo

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 bla bla bla bla bla bla bla bla bla bla.

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 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.

Indice analitico

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 bla bla bla bla bla bla bla bla bla bla.

55.2 Caratteri speciali

La realizzazione di un documento HTML standard richiede l'uso di una codifica dei caratteri limitata al codice ASCII tradizionale, dove in sostanza di hanno numeri, lettere non accentate e pochi altri simboli. Per tutti gli altri simboli, compresi quelli che nel sorgente HTML hanno un valore speciale (si tratta precisamente dei simboli '**<**', '**>**' e '**&**'), si usano delle rappresentazioni speciali, che qui vengono chiamate ***macro***.

Le macro che consentono di rappresentare caratteri speciali fanno riferimento a ***entità*** SGML; per esempio, '**`à`**' è la macro dell'entità '**`agrave`**'. Si osservi che le macro iniziano con la e-commerciale ('**&**') e terminano con il punto e virgola ('**;**').

Le tabelle successive riportano l'elenco completo delle macro SGML standard che si possono usare nei documenti HTML.

Tabella 55.12. Entità HTML HTMLlat1.

Codifica	Standard	SGML macro					Descrizione
U+00A0	ISOnum						no-break space = non-breaking space
U+00A1	ISOnum	¡	¡	¡	¡	¡	inverted exclamation mark
U+00A2	ISOnum	¢	¢	¢	¢	¢	cent sign
U+00A3	ISOnum	£	£	£	£	£	pound sign
U+00A4	ISOnum	¤	□	□	□	□	currency sign
U+00A5	ISOnum	¥	¥	¥	¥	¥	yen sign = yuan sign
U+00A6	ISOnum	¦	¦	¦	¦	¦	broken bar = broken vertical bar
U+00A7	ISOnum	§	§	§	§	§	section sign
U+00A8	ISOdia	¨	¨	¨	¨	¨	diaeresis = spacing diaeresis
U+00A9	ISOnum	©	©	©	©	©	copyright sign
U+00AA	ISOnum	ª	ª	ª	ª	ª	feminine ordinal indicator
U+00AB	ISOnum	«	«	«	«	«	left-pointing double angle quotation mark = left pointing guillemet
U+00AC	ISOnum	¬	¬	¬	¬	¬	not sign
U+00AD	ISOnum						soft hyphen = discretionary hyphen
U+00AE	ISOnum	®	®	®	®	®	registered sign = registered trade mark sign
U+00AF	ISOdia	¯	¯	¯	¯	¯	macron = spacing macron = overline = APL overbar
U+00B0	ISOnum	°	°	°	°	°	degree sign
U+00B1	ISOnum	±	±	±	±	±	plus-minus sign = plus-or-minus sign
U+00B2	ISOnum	²	²	²	²	²	superscript two = superscript digit two = squared
U+00B3	ISOnum	³	³	³	³	³	superscript three = superscript digit three = cubed
U+00B4	ISOdia	´	´	´	´	´	acute accent = spacing acute
U+00B5	ISOnum	µ	µ	µ	µ	µ	micro sign
U+00B6	ISOnum	¶	¶	¶	¶	¶	pilcrow sign = paragraph sign
U+00B7	ISOnum	·	·	·	·	·	middle dot = Georgian comma = Greek middle dot
U+00B8	ISOdia	¸	¸	¸	¸	¸	cedilla = spacing cedilla
U+00B9	ISOnum	¹	¹	¹	¹	¹	superscript one = superscript digit one
U+00BA	ISOnum	º	º	º	º	º	masculine ordinal indicator
U+00BB	ISOnum	»	»	»	»	»	right-pointing double angle quotation mark = right pointing guillemet
U+00BC	ISOnum	¼	¼	¼	¼	¼	vulgar fraction one quarter = fraction one quarter
U+00BD	ISOnum	½	½	½	½	½	vulgar fraction one half = fraction one half
U+00BE	ISOnum	¾	¾	¾	¾	¾	vulgar fraction three quarters = fraction three quarters

Codifica	Standard	SGML macro					Descrizione
U+00BF	ISOnum	¿	¿	*¿*	**¿**	**¿**	inverted question mark = turned question mark
U+00C0	ISOlat1	À	À	*À*	**À**	**À**	latin capital letter A with grave = latin capital letter A grave
U+00C1	ISOlat1	Á	Á	*Á*	**Á**	**Á**	latin capital letter A with acute
U+00C2	ISOlat1	Â	Â	*Â*	**Â**	**Â**	latin capital letter A with circumflex
U+00C3	ISOlat1	Ã	Ã	*Ã*	**Ã**	**Ã**	latin capital letter A with tilde
U+00C4	ISOlat1	Ä	Ä	*Ä*	**Ä**	**Ä**	latin capital letter A with diaeresis
U+00C5	ISOlat1	Å	Å	*Å*	**Å**	**Å**	latin capital letter A with ring above = latin capital letter A ring
U+00C6	ISOlat1	Æ	Æ	*Æ*	**Æ**	**Æ**	latin capital letter AE = latin capital ligature AE
U+00C7	ISOlat1	Ç	Ç	*Ç*	**Ç**	**Ç**	latin capital letter C with cedilla
U+00C8	ISOlat1	È	È	*È*	**È**	**È**	latin capital letter E with grave
U+00C9	ISOlat1	É	É	*É*	**É**	**É**	latin capital letter E with acute
U+00CA	ISOlat1	Ê	Ê	*Ê*	**Ê**	**Ê**	latin capital letter E with circumflex
U+00CB	ISOlat1	Ë	Ë	*Ë*	**Ë**	**Ë**	latin capital letter E with diaeresis
U+00CC	ISOlat1	Ì	Ì	*Ì*	**Ì**	**Ì**	latin capital letter I with grave
U+00CD	ISOlat1	Í	Í	*Í*	**Í**	**Í**	latin capital letter I with acute
U+00CE	ISOlat1	Î	Î	*Î*	**Î**	**Î**	latin capital letter I with circumflex
U+00CF	ISOlat1	Ï	Ï	*Ï*	**Ï**	**Ï**	latin capital letter I with diaeresis
U+00D0	ISOlat1	Ð	Ð	*Ð*	**Ð**	**Ð**	latin capital letter ETH
U+00D1	ISOlat1	Ñ	Ñ	*Ñ*	**Ñ**	**Ñ**	latin capital letter N with tilde
U+00D2	ISOlat1	Ò	Ò	*Ò*	**Ò**	**Ò**	latin capital letter O with grave
U+00D3	ISOlat1	Ó	Ó	*Ó*	**Ó**	**Ó**	latin capital letter O with acute
U+00D4	ISOlat1	Ô	Ô	*Ô*	**Ô**	**Ô**	latin capital letter O with circumflex
U+00D5	ISOlat1	Õ	Õ	*Õ*	**Õ**	**Õ**	latin capital letter O with tilde
U+00D6	ISOlat1	Ö	Ö	*Ö*	**Ö**	**Ö**	latin capital letter O with diaeresis
U+00D7	ISOnum	×	×	×	×	×	multiplication sign
U+00D8	ISOlat1	Ø	Ø	*Ø*	**Ø**	**Ø**	latin capital letter O with stroke = latin capital letter O slash
U+00D9	ISOlat1	Ù	Ù	*Ù*	**Ù**	**Ù**	latin capital letter U with grave
U+00DA	ISOlat1	Ú	Ú	*Ú*	**Ú**	**Ú**	latin capital letter U with acute
U+00DB	ISOlat1	Û	Û	*Û*	**Û**	**Û**	latin capital letter U with circumflex
U+00DC	ISOlat1	Ü	Ü	*Ü*	**Ü**	**Ü**	latin capital letter U with diaeresis
U+00DD	ISOlat1	Ý	Ý	*Ý*	**Ý**	**Ý**	latin capital letter Y with acute
U+00DE	ISOlat1	Þ	Þ	*Þ*	**Þ**	**Þ**	latin capital letter THORN
U+00DF	ISOlat1	ß	ß	*ß*	**ß**	**ß**	latin small letter sharp s = ess-zed
U+00E0	ISOlat1	à	à	*à*	**à**	**à**	latin small letter a with grave = latin small letter a grave
U+00E1	ISOlat1	á	á	*á*	**á**	**á**	latin small letter a with acute
U+00E2	ISOlat1	â	â	*â*	**â**	**â**	latin small letter a with circumflex
U+00E3	ISOlat1	ã	ã	*ã*	**ã**	**ã**	latin small letter a with tilde
U+00E4	ISOlat1	ä	ä	*ä*	**ä**	**ä**	latin small letter a with diaeresis
U+00E5	ISOlat1	å	å	*å*	**å**	**å**	latin small letter a with ring above = latin small letter a ring
U+00E6	ISOlat1	æ	æ	*æ*	**æ**	**æ**	latin small letter ae = latin small ligature ae
U+00E7	ISOlat1	ç	ç	*ç*	**ç**	**ç**	latin small letter c with cedilla
U+00E8	ISOlat1	è	è	*è*	**è**	**è**	latin small letter e with grave
U+00E9	ISOlat1	é	é	*é*	**é**	**é**	latin small letter e with acute
U+00EA	ISOlat1	ê	ê	*ê*	**ê**	**ê**	latin small letter e with circumflex
U+00EB	ISOlat1	ë	ë	*ë*	**ë**	**ë**	latin small letter e with diaeresis
U+00EC	ISOlat1	ì	ì	*ì*	**ì**	**ì**	latin small letter i with grave
U+00ED	ISOlat1	í	í	*í*	**í**	**í**	latin small letter i with acute
U+00EE	ISOlat1	î	î	*î*	**î**	**î**	latin small letter i with circumflex
U+00EF	ISOlat1	ï	ï	*ï*	**ï**	**ï**	latin small letter i with diaeresis
U+00F0	ISOlat1	ð	ð	*ð*	**ð**	**ð**	latin small letter eth
U+00F1	ISOlat1	ñ	ñ	*ñ*	**ñ**	**ñ**	latin small letter n with tilde
U+00F2	ISOlat1	ò	ò	*ò*	**ò**	**ò**	latin small letter o with grave
U+00F3	ISOlat1	ó	ó	*ó*	**ó**	**ó**	latin small letter o with acute
U+00F4	ISOlat1	ô	ô	*ô*	**ô**	**ô**	latin small letter o with circumflex
U+00F5	ISOlat1	õ	õ	*õ*	**õ**	**õ**	latin small letter o with tilde
U+00F6	ISOlat1	ö	ö	*ö*	**ö**	**ö**	latin small letter o with diaeresis
U+00F7	ISOnum	÷	÷	÷	÷	÷	division sign
U+00F8	ISOlat1	ø	ø	*ø*	**ø**	**ø**	latin small letter o with stroke = latin small letter o slash
U+00F9	ISOlat1	ù	ù	*ù*	**ù**	**ù**	latin small letter u with grave
U+00FA	ISOlat1	ú	ú	*ú*	**ú**	**ú**	latin small letter u with acute
U+00FB	ISOlat1	û	û	*û*	**û**	**û**	latin small letter u with circumflex
U+00FC	ISOlat1	ü	ü	*ü*	**ü**	**ü**	latin small letter u with diaeresis
U+00FD	ISOlat1	ý	ý	*ý*	**ý**	**ý**	latin small letter y with acute
U+00FE	ISOlat1	þ	þ	*þ*	**þ**	**þ**	latin small letter thorn
U+00FF	ISOlat1	ÿ	ÿ	*ÿ*	**ÿ**	**ÿ**	latin small letter y with diaeresis

Tabella 55.13. Entità HTML HTMLspecial.

Codifica	Standard	SGML macro					Descrizione
U+0022	ISOnum	"	"	*″*	**"**	**"**	quotation mark = APL quote
U+0026	ISOnum	&	&	*&*	**&**	**&**	ampersand
U+003C	ISOnum	<	<	*<*	**<**	**<**	less-than sign
U+003E	ISOnum	>	>	*>*	**>**	**>**	greater-than sign
U+0152	ISOlat2	Œ	Œ	*Œ*	**Œ**	**Œ**	latin capital ligature OE
U+0153	ISOlat2	œ	œ	*œ*	**œ**	**œ**	latin small ligature oe
U+0160	ISOlat2	Š	Š	*Š*	**Š**	**Š**	latin capital letter S with caron
U+0161	ISOlat2	š	š	*š*	**š**	**š**	latin small letter s with caron
U+0178	ISOlat2	Ÿ	Ÿ	*Ÿ*	**Ÿ**	**Ÿ**	latin capital letter Y with diaeresis

Codifica	Standard	SGML macro					Descrizione
U+02C6	ISOpub	ˆ	^	^	^	^	modifier letter circumflex accent
U+02DC	ISOdia	˜	~	~	~	~	small tilde
U+2002	ISOpub						en space
U+2003	ISOpub						em space
U+2009	ISOpub						thin space
U+200C	RFC 2070	‌					zero width non-joiner
U+200D	RFC 2070	‍					zero width joiner
U+200E	RFC 2070	‎	□	□	□	□	left-to-right mark
U+200F	RFC 2070	‏	□	□	□	□	right-to-left mark
U+2013	ISOpub	–	–	–	–	-	en dash
U+2014	ISOpub	—	—	—	—	--	em dash
U+2018	ISOnum	‘	‘	‘	‘	’	left single quotation mark
U+2019	ISOnum	’	’	’	’	’	right single quotation mark
U+201A		‚	’	’	’	’	single low-9 quotation mark
U+201C	ISOnum	“	“	“	“	”	left double quotation mark
U+201D	ISOnum	”	”	”	”	”	right double quotation mark
U+201E		„	”	”	”	”	double low-9 quotation mark
U+2020	ISOpub	†	†	†	†	†	dagger
U+2021	ISOpub	‡	‡	‡	‡	‡	double dagger
U+2030	ISOtech	‰	‰	‰	‰	‰	per mille sign
U+2039	ISO proposed	‹	⟨	⟨	⟨	⟨	single left-pointing angle quotation mark
U+203A	ISO proposed	›	⟩	⟩	⟩	⟩	single right-pointing angle quotation mark
U+20AC		€	€	€	€	€	euro sign

Tabella 55.14. Entità HTML HTMLsymbol.

Codifica	Standard	SGML macro					Descrizione
U+0192	ISOtech	ƒ	*f*	*f*	*f*	*f*	latin small f with hook = function= florin
U+0391		Α	A	A	A	A	greek capital letter alpha
U+0392		Β	B	B	B	B	greek capital letter beta
U+0393	ISO-grk3	Γ	Γ	Γ	Γ	Γ	greek capital letter gamma
U+0394	ISO-grk3	Δ	Δ	Δ	Δ	Δ	greek capital letter delta
U+0395		Ε	E	E	E	E	greek capital letter epsilon
U+0396		Ζ	Z	Z	Z	Z	greek capital letter zeta
U+0397		Η	H	H	H	H	greek capital letter eta
U+0398	ISO-grk3	Θ	Θ	Θ	Θ	Θ	greek capital letter theta
U+0399		Ι	I	I	I	I	greek capital letter iota
U+039A		Κ	K	K	K	K	greek capital letter kappa
U+039B	ISO-grk3	Λ	Λ	Λ	Λ	Λ	greek capital letter lambda
U+039C		Μ	M	M	M	M	greek capital letter mu
U+039D		Ν	N	N	N	N	greek capital letter nu
U+039E	ISO-grk3	Ξ	Ξ	Ξ	Ξ	Ξ	greek capital letter xi
U+039F		Ο	O	O	O	O	greek capital letter omicron

Codifica	Standard	SGML macro					Descrizione
U+03A0	ISO-grk3	Π	Π	Π	Π	Π	greek capital letter pi
U+03A1		Ρ	P	P	P	P	greek capital letter rho
U+03A3	ISO-grk3	Σ	Σ	Σ	Σ	Σ	greek capital letter sigma
U+03A4		Τ	T	T	T	T	greek capital letter tau
U+03A5	ISO-grk3	Υ	Υ	Υ	Υ	Υ	greek capital letter upsilon
U+03A6	ISO-grk3	Φ	Φ	Φ	Φ	Φ	greek capital letter phi
U+03A7		Χ	X	X	X	X	greek capital letter chi
U+03A8	ISO-grk3	Ψ	Ψ	Ψ	Ψ	Ψ	greek capital letter psi
U+03A9	ISO-grk3	Ω	Ω	Ω	Ω	Ω	greek capital letter omega
U+03B1	ISO-grk3	α	*α*	*α*	*α*	*α*	greek small letter alpha
U+03B2	ISO-grk3	β	*β*	*β*	*β*	*β*	greek small letter beta
U+03B3	ISO-grk3	γ	*γ*	*γ*	*γ*	*γ*	greek small letter gamma
U+03B4	ISO-grk3	δ	*δ*	*δ*	*δ*	*δ*	greek small letter delta
U+03B5	ISO-grk3	ε	*ϵ*	*ϵ*	*ϵ*	*ϵ*	greek small letter epsilon
U+03B6	ISO-grk3	ζ	*ζ*	*ζ*	*ζ*	*ζ*	greek small letter zeta
U+03B7	ISO-grk3	η	*η*	*η*	*η*	*η*	greek small letter eta
U+03B8		θ	*θ*	*θ*	*θ*	*θ*	greek small letter theta
U+03B9	ISO-grk3	ι	*ι*	*ι*	*ι*	*ι*	greek small letter iota
U+03BA	ISO-grk3	κ	*κ*	*κ*	*κ*	*κ*	greek small letter kappa
U+03BB	ISO-grk3	λ	*λ*	*λ*	*λ*	*λ*	greek small letter lambda
U+03BC	ISO-grk3	μ	*μ*	*μ*	*μ*	*μ*	greek small letter mu
U+03BD	ISO-grk3	ν	*ν*	*ν*	*ν*	*ν*	greek small letter nu
U+03BE	ISO-grk3	ξ	*ξ*	*ξ*	*ξ*	*ξ*	greek small letter xi
U+03BF		ο	o	o	o	o	greek small letter omicron
U+03C0	ISO-grk3	π	*π*	*π*	*π*	*π*	greek small letter pi
U+03C1	ISO-grk3	ρ	*ρ*	*ρ*	*ρ*	*ρ*	greek small letter rho
U+03C2	ISO-grk3	ς	*ς*	*ς*	*ς*	*ς*	greek small letter final sigma
U+03C3	ISO-grk3	σ	*σ*	*σ*	*σ*	*σ*	greek small letter sigma
U+03C4	ISO-grk3	τ	*τ*	*τ*	*τ*	*τ*	greek small letter tau
U+03C5	ISO-grk3	υ	*υ*	*υ*	*υ*	*υ*	greek small letter upsilon
U+03C6		φ	*ϕ*	*ϕ*	*ϕ*	*ϕ*	greek small letter phi
U+03C7	ISO-grk3	χ	*χ*	*χ*	*χ*	*χ*	greek small letter chi
U+03C8	ISO-grk3	ψ	*ψ*	*ψ*	*ψ*	*ψ*	greek small letter psi
U+03C9	ISO-grk3	ω	*ω*	*ω*	*ω*	*ω*	greek small letter omega
U+03D1		ϑ	*ϑ*	*ϑ*	*ϑ*	*ϑ*	greek small letter theta symbol
U+03D2		ϒ	Υ	Υ	Υ	Υ	greek upsilon with hook symbol
U+03D6	ISO-grk3	ϖ	*ϖ*	*ϖ*	*ϖ*	*ϖ*	greek pi symbol
U+2022	ISOpub	•	•	•	•	•	bullet = black small circle

Codifica	Standard	SGML macro					Descrizione
U+2026	ISOpub	…	…	…	…	…	horizontal ellipsis = three dot leader
U+2032	ISOtech	′	′	′	′	′	prime = minutes = feet
U+2033	ISOtech	″	″	″	″	″	double prime = seconds = inches
U+203E		‾	‾	‾	‾	‾	overline = spacing overscore
U+2044		⁄	⁄	⁄	⁄	⁄	fraction slash
U+2118	ISOamso	℘	℘	℘	℘	℘	script capital P = power set = Weierstrass p
U+2111	ISOamso	ℑ	ℑ	ℑ	ℑ	ℑ	blackletter capital I = imaginary part
U+211C	ISOamso	ℜ	ℜ	ℜ	ℜ	ℜ	blackletter capital R = real part symbol
U+2122	ISOnum	™	™	™	™	™	trade mark sign
U+2135		ℵ	ℵ	ℵ	ℵ	ℵ	alef symbol = first transfinite cardinal
U+2190	ISOnum	←	←	←	←	←	leftwards arrow
U+2191	ISOnu	↑	↑	↑	↑	↑	upwards arrow
U+2192	ISOnum	→	→	→	→	→	rightwards arrow
U+2193	ISOnum	↓	v	v	**v**	**v**	downwards arrow
U+2194	ISOamsa	↔	↔	↔	↔	↔	left right arrow
U+21B5		↵	↵	↵	↵	↵	downwards arrow with corner leftwards = carriage return
U+21D0	ISOtech	⇐	⇐	⇐	⇐	⇐	leftwards double arrow
U+21D1	ISOamsa	⇑	⇑	⇑	⇑	⇑	upwards double arrow
U+21D2	ISOtech	⇒	⇒	⇒	⇒	⇒	rightwards double arrow
U+21D3	ISOamsa	⇓	⇓	⇓	⇓	⇓	downwards double arrow
U+21D4	ISOamsa	⇔	⇔	⇔	⇔	⇔	left right double arrow
U+2200	ISOtech	∀	∀	∀	∀	∀	for all
U+2202	ISOtech	∂	∂	∂	∂	∂	partial differential
U+2203	ISOtech	∃	∃	∃	∃	∃	there exists
U+2205	ISOamso	∅	∅	∅	∅	∅	empty set = null set = diameter
U+2207	ISOtech	∇	∇	∇	∇	∇	nabla = backward difference
U+220A	ISOtech	∈	□	□	□	□	element of
U+2209	ISOtech	∉	∉	∉	∉	∉	not an element of
U+220D	ISOtech	∋	□	□	□	□	contains as member
U+220F	ISOamsb	∏	∏	∏	∏	∏	n-ary product = product sign
U+2211	ISOamsb	∑	∑	∑	∑	∑	n-ary sumation
U+2212	ISOtech	−	−	_	−	−	minus sign
U+2217	ISOtech	∗	*	*	*	*	asterisk operator
U+221A	ISOtech	√	√	√	√	√	square root = radical sign
U+221D	ISOtech	∝	∝	∝	∝	∝	proportional to
U+221E	ISOtech	∞	∞	∞	∞	∞	infinity
U+2220	ISOamso	∠	∠	∠	∠	∠	angle
U+2227	ISOtech	∧	∧	∧	∧	∧	logical and = wedge
U+2228	ISOtech	∨	∨	∨	∨	∨	logical or = vee
U+2229	ISOtech	∩	∩	∩	∩	∩	intersection = cap
U+222A	ISOtech	∪	∪	∪	∪	∪	union = cup
U+222B	ISOtech	∫	∫	∫	∫	∫	integral
U+2234	ISOtech	∴	∴	∴	∴	∴	therefore
U+223C	ISOtech	∼	~	~	~	~	tilde operator = varies with = similar to
U+2245	ISOtech	≅	≅	≅	≅	≅	approximately equal to

Codifica	Standard	SGML macro					Descrizione
U+2248	ISOamsr	≈	≍	≍	≍	≍	almost equal to = asymptotic to
U+2260	ISOtech	≠	≠	≠	≠	≠	not equal to
U+2261	ISOtech	≡	≡	≡	≡	≡	identical to
U+2264	ISOtech	≤	≤	≤	≤	≤	less-than or equal to
U+2265	ISOtech	≥	≥	≥	≥	≥	greater-than or equal to
U+2282	ISOtech	⊂	⊂	⊂	⊂	⊂	subset of
U+2283	ISOtech	⊃	⊃	⊃	⊃	⊃	superset of
U+2284	ISOamsn	⊄	⊄	⊄	⊄	⊄	not a subset of
U+2286	ISOtech	⊆	⊆	⊆	⊆	⊆	subset of or equal to
U+2287	ISOtech	⊇	⊇	⊇	⊇	⊇	superset of or equal to
U+2295	ISOamsb	⊕	⊕	⊕	⊕	⊕	circled plus = direct sum
U+2297	ISOamsb	⊗	⊗	⊗	⊗	⊗	circled times = vector product
U+22A5	ISOtech	⊥	⊥	⊥	⊥	⊥	up tack = orthogonal to = perpendicular
U+22C5	ISOamsb	⋅	·	·	·	·	dot operator
U+2308	ISOamsc	⌈	⌈	⌈	⌈	⌈	left ceiling = apl upstile
U+2309	ISOamsc	⌉	⌉	⌉	⌉	⌉	right ceiling
U+230A	ISOamsc	⌊	⌊	⌊	⌊	⌊	left floor = apl downstile
U+230B	ISOamsc	⌋	⌋	⌋	⌋	⌋	right floor
U+2329	ISOtech	⟨	□	□	□	□	left-pointing angle bracket = bra
U+232A	ISOtech	⟩	□	□	□	□	right-pointing angle bracket = ket
U+25CA	ISOpub	◊	◊	◊	◊	◊	lozenge
U+2660	ISOpub	♠	♠	♠	♠	♠	black spade suit
U+2663	ISOpub	♣	♣	♣	♣	♣	black club suit = shamrock
U+2665	ISOpub	♥	♥	♥	♥	♥	black heart suit = valentine
U+2666	ISOpub	♦	♦	♦	♦	♦	black diamond suit

Oltre alle macro delle entità standard, descritte nelle tabelle appena mostrate, è possibile usare altre macro, che fanno riferimento al codice numerico dei caratteri che si vogliono usare. Esistono due modi alternativi: '**`&#n;`**' oppure '**`&#xn;`**'. Per esempio, '**`&#232;`**' oppure '**`&#xe8;`**' rappresentano indifferentemente la lettera «è». La differenza sta nel fatto che nel primo caso, si usa il codice numerico in base dieci, mentre nel secondo si usa il codice in esadecimale (in base sedici). Le tabelle delle macro mostrano nella prima colonna il valore dell'insieme di caratteri universale, che si esprime con un numero in esadecimale.

La sigla esadecimale da usare nelle macro del tipo '**`&#xn;`**' deve essere composta con lettere minuscole. La sigla indicata nella prima colonna delle tabelle contiene il prefisso «U+» che deve essere rimosso nella rappresentazione della macro.

In generale, se possibile, è bene usare le macro standard che, attraverso il loro nome, ricordano anche di che carattere si tratta, mentre per poter scrivere altri simboli mancanti occorre usare necessariamente le macro numeriche.

55.2.1 File «macro-01.html»

Si crei il file '`macro-01.html`', sfruttando eventualmente una copia di un altro file HTML creato in precedenza, con il contenuto seguente:

```
1  <!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
2  <HTML LANG="it">
3  <HEAD>
4      <META HTTP-EQUIV="Content-Type"
```

```
5           CONTENT="text/html; charset=us-ascii">
6       <META NAME="Description"
7           CONTENT="Macro HTML, esempio 01">
8       <META NAME="Keywords"
9           CONTENT="HTML, macro, entit&agrave;, esempio">
10      <META NAME="Author"
11          CONTENT="Tizio Tizi, tizio@brot.dg, classe 4Z">
12      <META NAME="Date" CONTENT="2012.01.01">
13      <META NAME="Resource-type" LANG="en"
14          CONTENT="Document">
15      <META NAME="Revisit-after" LANG="en"
16          CONTENT="15 days">
17      <META NAME="Robots" CONTENT="ALL">
18      <TITLE>Macro HTML</TITLE>
19  </HEAD>
20  <BODY>
21
22  <H1>Entit&agrave; SGML utilizzate nei documenti HTML</H1>
23
24  <P>Un file HTML pu&ograve; contenere delle macro
25  speciali, composte da un nome, circoscritto
26  inizialmente dal simbolo &amp; e concluso da un punto e
27  virgola.</P>
28
29  <P>Le macro servono per poter inserire simboli come
30  &lt; e &gt;, che vengono usati nel sorgente HTML con un
31  significato speciale, per le lettere accentate degli
32  alfabeti latini e per la rappresentazione di alfabeti
33  esotici.</P>
34
35  <P>Per la lingua italiana servono in modo particolare
36  le vocali accentate: &agrave;, &egrave;, &igrave;,
37  &ograve;, &ugrave;, &eacute;, &Agrave;, &Egrave;,
38  &Igrave;,
39  &Ograve;, &Ugrave;, &Eacute;.</p>
40
41  <P>Quando si usano le macro all'interno degli attributi
42  di un elemento, che sono gi&agrave; delimitati da apici
43  doppi, questi si possono ottenere con la macro
44  &amp;quot; (&quot;).</P>
45
46  </BODY>
47  </HTML>
```

Come sempre, si deve verificare la correttezza sintattica e quindi si deve verificare il risultato visto da un navigatore.

Figura 55.16. Aspetto del file ‘`macro-01.html`’ con un navigatore.

Entità SGML utilizzate nei documenti HTML

Un file HTML può contenere delle macro speciali, composte da un nome, circoscritto inizialmente dal simbolo & e concluso da un punto e virgola.

Le macro servono per poter inserire simboli come < e >, che vengono usati nel sorgente HTML con un significato speciale, per le lettere accentate degli alfabeti latini e per la rappresentazione di alfabeti esotici.

Per la lingua italiana servono in modo particolare le vocali accentate: à, è, ì, ò, ù, é, À, È, Ì, Ò, Ù, É.

Quando si usano le macro all'interno degli attributi di un elemento, che sono già delimitati da apici doppi, questi si possono ottenere con la macro " (").

55.2.2 File «macro-02.html»

Per esercizio, si salvi il file con il nome ‘`macro-02.html`’, modificando tutte le macro che fanno riferimento a entità standard, con macro numeriche, secondo la forma ‘**`&#xn;`**’ (con il valore in esadecimale). Per fare questo si devono consultare le tabelle che appaiono all'inizio del capitolo.

Si controlli la correttezza sintattica e che il risultato visto dal navigatore non sia cambiato.

55.2.3 Verifica sulla struttura generale e sull'uso delle macro

In base a quanto appreso fino a questo punto, si realizzi il file ‘`verifica-struttura-macro.html`’ che produca un risultato abbastanza simile a quello seguente, quando viene visto attraverso un navigatore:

Cappuccetto Rosso

di Jakob e Wilhelm Grimm

C'era una volta una cara ragazzina; solo a vederla le volevan tutti bene, e specialmente la nonna, che non sapeva più cosa regalarle. Una volta le regalò un cappuccetto di velluto rosso, e, poiché le donava tanto ch'essa non volle più portare altro, la chiamarono sempre Cappuccetto Rosso.

[...]

Si completi l'intestazione, coerentemente, secondo il proprio criterio, seguendo le modalità già stabilite. Si deve consegnare per la valutazione:

1. la stampa del risultato ottenuto attraverso il navigatore;
2. la stampa del sorgente.

Si richiede espressamente che tutti gli elementi, a esclusione di quelli che devono essere vuoti (come il caso di ‘**`META`**’), siano terminati correttamente con il marcatore di chiusura.

55.3 Contenuto del testo lineare

Il testo lineare è composto da caratteri tipografici, macro che si espandono in caratteri tipografici particolari ed elementi che delimitano porzioni del testo o che servono a inserire oggetti speciali nel testo. La tabella seguente elenca gli elementi più comuni per delimitare e qualificare delle porzioni di testo, oppure per interromperlo.

Tabella 55.18. Elementi da usare all'interno delle frasi.

Elemento	Significato
EM	Testo enfatizzato, di solito in corsivo.
STRONG	Testo evidenziato, di solito in neretto.
CITE	Citazione, nel senso di chi o cosa viene citato.
Q	Testo citato.
DFN	Definizione.
CODE	Codice usato in elaborazione, di solito reso in carattere dattilografico.
SAMP	Testo risultato di un'elaborazione.
KBD	Testo da inserire attraverso la tastiera.
VAR	Variabile o argomento di un programma.
ABBR	Abbreviazione.
ACRONYM	Acronimo.
SUB	Testo a pedice.
SUP	Testo ad apice.
BR	Va a capo (elemento vuoto).

Questi elementi possono essere usati con l'attributo ‘**`TITLE`**’, con il quale si dà un titolo al contenuto. Per esempio, l'elemento ‘**`ACRONYM`**’ può essere usato per circoscrivere un acronimo, ovvero una sigla, mentre con l'attributo ‘**`TITLE`**’ si va a descrivere in modo esteso il suo significato:

```
<P>Il <ACRONYM TITLE="World Wide Web
Consortium">W3C</ACRONYM> è un consorzio per lo
sviluppo di tecnologie di...</P>
```

55.3.1 File «testo-lineare-01.html»

Si crei il file ‘`testo-lineare-01.html`’, sfruttando eventualmente una copia di un altro file HTML creato in precedenza, con il contenuto seguente:

```
1   <!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
2   <HTML LANG="it">
3   <HEAD>
4       <META HTTP-EQUIV="Content-Type"
5           CONTENT="text/html; charset=us-ascii">
6       <META NAME="Description" CONTENT="Elementi tipici
7       contenuti nel testo lineare, esempio 01">
8       <META NAME="Keywords" CONTENT="HTML, testo lineare,
9       elementi, esempio">
10      <META NAME="Author"
11          CONTENT="Tizio Tizi, tizio@brot.dg, classe 4Z">
12      <META NAME="Date" CONTENT="2012.01.01">
13      <META NAME="Resource-type" LANG="en"
14          CONTENT="Document">
```

```
    <META NAME="Revisit-after" LANG="en"
          CONTENT="15 days">
    <META NAME="Robots" CONTENT="ALL">
    <TITLE>Elementi del testo lineare</TITLE>
</HEAD>
<BODY>

<H1>Elementi del testo lineare</H1>

<P>All'interno di un <EM>testo lineare</EM> si possono
usare elementi che delimitano e qualificano il testo
che racchiudono.<BR>Questo &egrave; un
<STRONG>neretto</STRONG>, questa &egrave; la molecola
dell'acqua: <CODE>H<SUB>2</SUB>O</CODE>.</P>

<P>Si osservi comunque che non tutti gli elementi che
delimitano del testo lineare producono necessariamente
un effetto visibile nel risultato ottenuto attraverso
il navigatore, oppure possono prevedere l'inserzione di
altri simboli. Per esempio, Dante Alighieri scrisse: <Q
TITLE="Inferno, canto I">Nel mezzo del cammin di nostra
vita mi ritrovai per una selva oscura ch&eacute; la
diritta via era smarrita.</Q></P>

</BODY>
</HTML>
```

Figura 55.21. Aspetto del file '`testo-lineare-01.html`' con un navigatore.

Elementi del testo lineare

All'interno di un *testo lineare* si possono usare elementi che delimitano e qualificano il testo che racchiudono.
Questo è un **neretto**, questa è la molecola dell'acqua: H_2O.

Si osservi comunque che non tutti gli elementi che delimitano del testo lineare producono necessariamente un effetto visibile nel risultato ottenuto attraverso il navigatore, oppure possono prevedere l'inserzione di altri simboli. Per esempio, Dante Alighieri scrisse: «Nel mezzo del cammin di nostra vita mi ritrovai per una selva oscura ché la diritta via era smarrita.»

Nel caso particolare dell'elemento '**`Q`**', dal punto di vista visivo, il navigatore dovrebbe inserire le virgolette appropriate, secondo il tipo di linguaggio a cui si sta facendo riferimento; inoltre, portando il puntatore del mouse sopra il testo delimitato dall'elemento '**`Q`**', dovrebbe apparire la nota contenuta nell'attributo '**`TITLE`**': «Inferno, canto I».

55.3.2 File «testo-lineare-02.html»

Per esercizio, si crei il file '`testo-lineare-02.html`', utilizzando tutti gli elementi che appaiono descritti nella tabella riassuntiva all'inizio del capitolo.

55.3.3 Verifica sull'uso degli elementi interni al testo lineare

In base a quanto appreso fino a questo punto, si realizzi il file '`verifica-testo-lineare-01.html`' che produca un risultato abbastanza simile a quello seguente, quando viene visto attraverso un navigatore:

Biancaneve

di Jakob e Wilhelm Grimm

C'era una volta una regina in attesa di un bambino che in una giornata invernale stava filando davanti alla finestra. Il davanzale era di legno d'ebano nero, e si stava ammucchiando già della neve. Ad un tratto si punse un dito ed alcune gocce di sangue caddero sulla neve. La regina pensò: «Come mi piacerebbe avere una bambina dai capelli neri come l'ebano, dalle labbra rosse come il sangue e dalla pelle bianca come la neve!»

Ma dopo poco si ammalò gravemente e morì. Per qualche anno il re suo marito fu inconsolabile: poi un giorno incontrò una bellissima dama a caccia, ricca e nobile e decise di sposarla per dare una mamma a Biancaneve. Ma ignorava che era in realtà una strega, esperta in pozioni magiche, con uno specchio magico a cui ogni giorno chiedeva: «Specchio, specchio delle mie brame, chi è la più bella del reame!» per il gusto di farsi rispondere «La più bella mia regina sei tu».

[...]

In corrispondenza del titolo, interno al corpo del documento, si vuole inserire l'anno di edizione (1812-1822), in modo che appaia quando vi si porta sopra il puntatore del mouse, usando il navigatore.

Si completi l'intestazione, coerentemente, secondo il proprio criterio, seguendo le modalità già stabilite. Si deve consegnare per la valutazione:

1. la stampa del risultato ottenuto attraverso il navigatore;
2. la stampa del sorgente.

Si richiede espressamente che tutti gli elementi, a esclusione di quelli che devono essere vuoti, siano terminati correttamente con il marcatore di chiusura.

55.3.4 Seconda verifica sull'uso degli elementi interni al testo lineare

In base a quanto appreso fino a questo punto, si realizzi il file '`verifica-testo-lineare-02.html`' che produca un risultato abbastanza simile a quello seguente, quando viene visto attraverso un navigatore:

Paradiso

di Dante Alighieri

Canto I

La gloria di colui che tutto move
per l'universo penetra, e risplende
in una parte più e meno altrove.

Nel ciel che più de la sua luce prende
fu' io, e vidi cose che ridire
né sa né può chi di là sù discende;

perché appressando sé al suo disire,
nostro intelletto si profonda tanto,
che dietro la memoria non può ire.

[...]

La verifica richiede in modo particolare l'uso appropriato degli elementi '**`P`**' e '**`BR`**'.

Si completi l'intestazione, coerentemente, secondo il proprio criterio, seguendo le modalità già stabilite. Si deve consegnare per la valutazione:

1. la stampa del risultato ottenuto attraverso il navigatore;
2. la stampa del sorgente.

Si richiede espressamente che tutti gli elementi, a esclusione di quelli che devono essere vuoti, siano terminati correttamente con il marcatore di chiusura.

55.4 Testo preformattato

Il carattere tipografico che si ottiene normalmente, visualizzando un documento HTML attraverso un navigatore grafico, ha una larghezza proporzionale. Quando c'è la necessità di rappresentare qualcosa con un carattere a larghezza uniforme, ovvero «dattilografico», si usano alcuni elementi appositi: '**`CODE`**' e '**`SAMP`**' nell'ambito di un testo lineare; '**`PRE`**' come blocco separato. L'elemento '**`PRE`**' in particolare è molto utile per la rappresentazione di listati, come può essere una porzione di un sorgente di un linguaggio di programmazione.

55.4.1 File «testo-preformattato-01.html»

Si crei il file '`testo-preformattato-01.html`' con il contenuto seguente:

```
<!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
<HTML LANG="it">
<HEAD>
    <META HTTP-EQUIV="Content-Type"
          CONTENT="text/html; charset=us-ascii">
    <META NAME="Description" CONTENT="Elementi usati
    per rappresentare testo dattilografico e listati,
    esempio 01">
```

```
    <META NAME="Keywords" CONTENT="HTML, testo
    preformattato, elementi, esempio">
    <META NAME="Author"
          CONTENT="Tizio Tizi, tizio@brot.dg, classe 4Z">
    <META NAME="Date" CONTENT="2012.01.01">
    <META NAME="Resource-type" LANG="en"
          CONTENT="Document">
    <META NAME="Revisit-after" LANG="en"
          CONTENT="15 days">
    <META NAME="Robots" CONTENT="ALL">
    <TITLE>Testo preformattato</TITLE>
</HEAD>
<BODY>

<H1>Linguaggio C</H1>

<P>Il codice di un programma C &egrave; scomposto in
funzioni, dove l'esecuzione del programma corrisponde
alla chiamata della funzione <CODE>main()</CODE>.
Questa funzione pu&ograve; essere dichiarata senza
argomenti oppure con due argomenti precisi: <SAMP>int
main (int argc, char *argv[])</SAMP>.</P>

<H2>Ciao mondo!</H2>

<P>Segue un esempio molto semplice di un programma che
emette un messaggio e poi termina la sua
esecuzione:</P>

<PRE>
#include &lt;stdio.h&gt;

int main ()
{
    printf ("Ciao mondo!\n");
}
</PRE>

<P>Dopo la compilazione del sorgente, si pu&ograve;
eseguire il programma eseguibile e il risultato
&egrave;: <SAMP>Ciao mondo!</SAMP></P>

</BODY>
</HTML>
```

Figura 55.25. Aspetto del file '`testo-preformattato-01.html`' con un navigatore.

Linguaggio C

Il codice di un programma C è scomposto in funzioni, dove l'esecuzione del programma corrisponde alla chiamata della funzione `main()`. Questa funzione può essere dichiarata senza argomenti oppure con due argomenti precisi: `int main (int argc, char *argv[])`.

Ciao mondo!

Segue un esempio molto semplice di un programma che emette un messaggio e poi termina la sua esecuzione:

```
#include <stdio.h>

int main ()
{
    printf ("Ciao mondo!\n");
}
```

Dopo la compilazione del sorgente, si può eseguire il programma eseguibile e il risultato è: `Ciao mondo!`

Come si può osservare, di solito non c'è differenza nel modo in cui viene reso tipograficamente il contenuto dell'elemento '**`CODE`**' rispetto a quello dell'elemento '**`SAMP`**'. Ciò che in particolare si deve comprendere dall'esempio, è il comportamento dell'elemento '**`PRE`**': è un blocco che contiene testo lineare, ma a differenza dell'elemento '**`P`**', gli spazi orizzontali e le interruzioni di riga vengono rispettati nella resa finale, così come sono nel sorgente.

55.4.2 File «testo-preformattato-02.html»

Si salvi il file nuovamente, ma con il nome '`testo-preformattato-02.html`', modificandone il contenuto nel modo seguente:


```
<!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
<HTML LANG="it">
<HEAD>
    <META HTTP-EQUIV="Content-Type"
          CONTENT="text/html; charset=us-ascii">
    <META NAME="Description" CONTENT="Elementi usati
    per rappresentare testo dattilografico e listati,
    esempio 02">
    <META NAME="Keywords" CONTENT="HTML, testo
    preformattato, elementi, esempio">
    <META NAME="Author"
          CONTENT="Tizio Tizi, tizio@brot.dg, classe 4Z">
    <META NAME="Date" CONTENT="2012.01.01">
    <META NAME="Resource-type" LANG="en"
          CONTENT="Document">
    <META NAME="Revisit-after" LANG="en"
          CONTENT="15 days">
    <META NAME="Robots" CONTENT="ALL">
    <TITLE>Testo preformattato</TITLE>
</HEAD>
<BODY>

<H1>Linguaggio C</H1>

<P>Il codice di un programma C &egrave; scomposto in
funzioni, dove l'esecuzione del programma corrisponde
alla chiamata della funzione <CODE>main()</CODE>.
Questa funzione pu&ograve; essere dichiarata senza
argomenti oppure con due argomenti precisi: <SAMP>int
main (int argc, char *argv[])</SAMP>.</P>

<H2>Ciao mondo!</H2>

<P>Segue un esempio molto semplice di un programma che
emette un messaggio e poi termina la sua
esecuzione:</P>

<P><CODE>
#include &lt;stdio.h&gt;<BR>
int main ()<BR>
{<BR>
    printf ("Ciao mondo!\n");<BR>
}
</CODE></P>

<P>Dopo la compilazione del sorgente, si pu&ograve;
eseguire il programma eseguibile e il risultato
&egrave;: <SAMP>Ciao mondo!</SAMP></P>

</BODY>
</HTML>
```

Come si può comprendere, si intende riprodurre il funzionamento dell'elemento '**`PRE`**' attraverso l'uso dell'elemento '**`CODE`**', inserendo degli elementi '**`BR`**' per interrompere esplicitamente le righe. Ciò che si ottiene assomiglia al file precedente, ma rimane irrisolto il problema degli spazi:

Figura 55.27. Aspetto del file '`testo-preformattato-02.html`' con un navigatore.

Linguaggio C

Il codice di un programma C è scomposto in funzioni, dove l'esecuzione del programma corrisponde alla chiamata della funzione `main()`. Questa funzione può essere dichiarata senza argomenti oppure con due argomenti precisi: `int main (int argc, char *argv[])`.

Ciao mondo!

Segue un esempio molto semplice di un programma che emette un messaggio e poi termina la sua esecuzione:

```
#include <stdio.h>
int main ()
{
printf ("Ciao mondo!\n");
}
```

Dopo la compilazione del sorgente, si può eseguire il programma eseguibile e il risultato è: `Ciao mondo!`

55.4.3 File «testo-preformattato-03.html»

Si salvi il file nuovamente, ma con il nome ‘`testo-preformattato-03.html`’, modificandone il contenuto nel modo seguente:

```
<!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
<HTML LANG="it">
<HEAD>
    <META HTTP-EQUIV="Content-Type"
          CONTENT="text/html; charset=us-ascii">
    <META NAME="Description" CONTENT="Elementi usati
    per rappresentare testo dattilografico e listati,
    esempio 03">
    <META NAME="Keywords" CONTENT="HTML, testo
    preformattato, elementi, esempio">
    <META NAME="Author"
          CONTENT="Tizio Tizi, tizio@brot.dg, classe 4Z">
    <META NAME="Date" CONTENT="2012.01.01">
    <META NAME="Resource-type" LANG="en"
          CONTENT="Document">
    <META NAME="Revisit-after" LANG="en"
          CONTENT="15 days">
    <META NAME="Robots" CONTENT="ALL">
    <TITLE>Testo preformattato</TITLE>
</HEAD>
<BODY>

<H1>Linguaggio C</H1>

<P>Il codice di un programma C &egrave; scomposto in
funzioni, dove l’esecuzione del programma corrisponde
alla chiamata della funzione <CODE>main()</CODE>.
Questa funzione pu&ograve; essere dichiarata senza
argomenti oppure con due argomenti precisi: <SAMP>int
main (int argc, char *argv[])</SAMP>.</P>

<H2>Ciao mondo!</H2>

<P>Segue un esempio molto semplice di un programma che
emette un messaggio e poi termina la sua
esecuzione:</P>

<P><CODE>
#include &lt;stdio.h&gt;<BR>
int main ()<BR>
{<BR>
    printf ("Ciao mondo!\n");<BR>
}
</CODE></P>

<P>Dopo la compilazione del sorgente, si pu&ograve;
eseguire il programma eseguibile e il risultato
&egrave;: <SAMP>Ciao mondo!</SAMP></P>

</BODY>
</HTML>
```

Anche in questo caso si vuole riprodurre il funzionamento dell’elemento ‘**PRE**’, ma questa volta si provvede anche a inserire degli spazi letterali (precisamente spazi non interrompibili), attraverso la macro ‘**` `**’ (*No-break space*). Il risultato è molto simile, visivamente, a quello dell’elemento ‘**PRE**’.

Figura 55.29. Aspetto del file ‘`testo-preformattato-03.html`’ con un navigatore.

Linguaggio C

Il codice di un programma C è scomposto in funzioni, dove l’esecuzione del programma corrisponde alla chiamata della funzione `main()`. Questa funzione può essere dichiarata senza argomenti oppure con due argomenti precisi: `int main (int argc, char *argv[])`.

Ciao mondo!

Segue un esempio molto semplice di un programma che emette un messaggio e poi termina la sua esecuzione:

```
#include <stdio.h>
int main ()
{
    printf ("Ciao mondo!\n");
}
```

Dopo la compilazione del sorgente, si può eseguire il programma eseguibile e il risultato è: `Ciao mondo!`

Sia chiaro, comunque, che quando è possibile e conveniente, è meglio usare l’elemento ‘**PRE**’.

55.4.4 Verifica sull’uso degli elementi per il testo preformattato

In base a quanto appreso fino a questo punto, si realizzi il file ‘`verifica-pre-01.html`’ che produca un risultato abbastanza simile a quello seguente, quando viene visto attraverso un navigatore. Si ricordi di usare l’elemento ‘**PRE**’ quando ciò è appropriato.

Java

I componenti del linguaggio Java hanno nomi diversi da quelli consueti. Volendo fare un abbinamento approssimativo con un linguaggio di programmazione normale, si potrebbe dire che in Java i programmi sono *classi* e le funzioni sono *metodi*. Naturalmente ci sono anche tante altre cose nuove.

Fatta questa premessa, si può dare un’occhiata alla solita classe banale: quella che visualizza un messaggio e termina.

```
/**
 *  CiaoMondoApp.java
 *  La solita classe banale.
 */

import java.lang.*; // predefinita

class CiaoMondoApp
{
    public static void main (String[] args)
    {
        System.out.println ("Ciao Mondo!"); // visualizza il messaggio
    }
}
```

Nel testo appaiono evidenziate le parole «classi» e «metodi», che sono da intendere come **definizioni**.

Si completi l’intestazione, coerentemente, secondo il proprio criterio, seguendo le modalità già stabilite. Si deve consegnare per la valutazione:

1. la stampa del risultato ottenuto attraverso il navigatore;
2. la stampa del sorgente.

Si richiede espressamente che tutti gli elementi, a esclusione di quelli che devono essere vuoti, siano terminati correttamente con il marcatore di chiusura.

55.5 Elenchi

Esistono tre tipi di elenchi: puntato, numerato e descrittivo. Gli elenchi puntati e quelli numerati condividono una struttura simile:

```
<UL>
<LI>
    …
</LI>
…
</UL>
```

```
<OL>
<LI>
    ...
</LI>
...
</OL>
```

Gli elenchi puntati sono delimitati dall'elemento '**UL**' (*Unordered list*), mentre quelli numerati dall'elemento '**OL**' (*Ordered list*). Questi elementi contengono uno o più elementi '**LI**', che possono contenere testo lineare, oppure uno o più blocchi (per esempio possono contenere dei sottoelenchi).

Figura 55.31. Struttura dell'elenco puntato con un esempio.

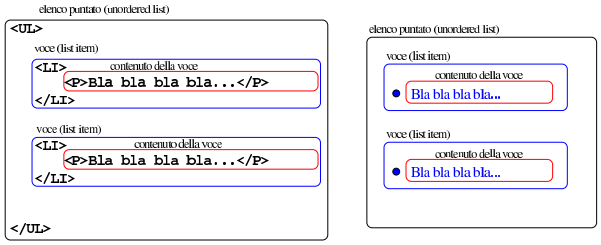
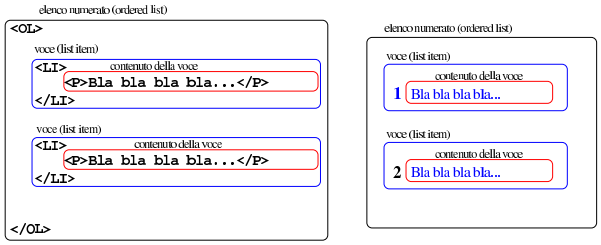


Figura 55.32. Struttura dell'elenco numerato con un esempio.



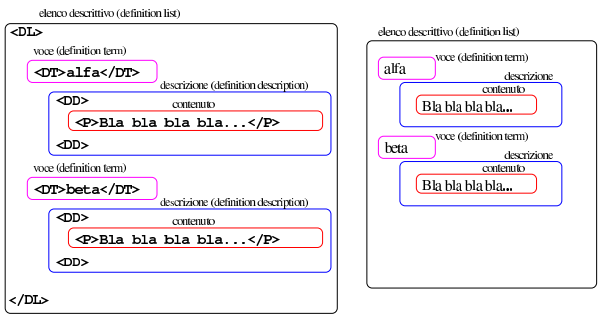
Per evitare ambiguità, conviene usare sempre dei blocchi per il contenuto degli elementi '**LI**', altrimenti, per l'abitudine, si rischia di pretendere di spezzare un testo lineare con un paragrafo o con un altro blocco che non può essere ammissibile in tale contesto.

Gli elenchi descrittivi sono più complessi e sono delimitati esternamente dall'elemento '**DL**' (*Definition list*), che contiene elementi '**DT**' (*Definition term*) e '**DD**' (*Definition description*):

```
<DL>
<DT>termine</DT>
    <DD>
        descrizione
    </DD>
    ...
</DL>
```

Il contenuto dell'elemento '**DT**' è del testo lineare, mentre '**DD**' può contenere sia testo lineare, sia dei blocchi. Anche in questo caso, per evitare ambiguità, è meglio usare sempre solo dei blocchi.

Figura 55.33. Struttura dell'elenco descrittivo con un esempio.



55.5.1 File «elenchi-01.html»

Si crei il file '`elenchi-01.html`' con il contenuto seguente:

```
<!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
<HTML LANG="it">
<HEAD>
    <META HTTP-EQUIV="Content-Type"
          CONTENT="text/html; charset=us-ascii">
    <META NAME="Description"
          CONTENT="Elenchi con HTML, esempio 01">
    <META NAME="Keywords"
          CONTENT="HTML, elenco, elenchi, esempio">
    <META NAME="Author"
          CONTENT="Tizio Tizi, tizio@brot.dg, classe 4Z">
    <META NAME="Date" CONTENT="2012.01.01">
    <META NAME="Resource-type" LANG="en"
          CONTENT="Document">
    <META NAME="Revisit-after" LANG="en"
          CONTENT="15 days">
    <META NAME="Robots" CONTENT="ALL">
    <TITLE>Elenchi</TITLE>
</HEAD>
<BODY>

<H1>Attivo</H1>

<DL>
<DT>A</DT>
    <DD>

        <P>Crediti verso i soci per versamenti ancora
        dovuti</P>

    </DD>
<DT>B</DT>
    <DD>

        <P>Immobilizzazioni</P>

        <UL>
        <LI>

            <P>I - Immobilizzazioni immateriali</P>

            <OL>
            <LI>

                <P>costi di impianto e di
                ampliamento</P>

            </LI>
            <LI>

                <P>costi di ricerca, di sviluppo e di
                pubblicit&agrave;</P>

            </LI>
            <LI>

                <P>diritti di brevetto industriale e
                diritti di utilizzazione delle opere
                dell'ingegno</P>

            </LI>
            <LI>

                <P>concessioni, licenze, marchi e
                diritti simili</P>

            </LI>
            <LI>

                <P>avviamento</P>

            </LI>
            <LI>

                <P>immobilizzazioni immateriali in
                corso e acconti</P>

            </LI>
            <LI>

                <P>altre immobilizzazioni
```

```
82                  immateriali</P>
83
84              </LI>
85              </OL>
86
87          </LI>
88          <LI>
89
90              <P>II - Immobilizzazioni materiali</P>
91
92              <OL>
93              <LI>
94
95                  <P>terreni e fabbricati</P>
96
97              </LI>
98              <LI>
99
100                 <P>[...]</P>
101
102             </LI>
103             </OL>
104
105         </LI>
106         <LI>
107
108             <P>[...]</P>
109
110         </LI>
111         </UL>
112
113     </DD>
114 </DL>
115
116 </BODY>
117 </HTML>
```

Il file ‘`elenchi-01.html`’ contiene un elenco descrittivo tra le righe 18 e 102. All’interno, in corrispondenza della seconda voce dell’elenco descrittivo, compare un elenco puntato (dalla riga 30 alla riga 99); infine, all’interno delle voci dell’elenco puntato appaiono degli elenchi numerati.

Figura 55.35. Aspetto del file ‘`elenchi-01.html`’ con un navigatore.

Attivo

A

Crediti verso i soci per versamenti ancora dovuti

B

Immobilizzazioni

- I - Immobilizzazioni immateriali
  1. costi di impianto e di ampliamento
  2. costi di ricerca, di sviluppo e di pubblicità
  3. diritti di brevetto industriale e diritti di utilizzazione delle opere dell’ingegno
  4. concessioni, licenze, marchi e diritti simili
  5. avviamento
  6. immobilizzazioni immateriali in corso e acconti
  7. altre immobilizzazioni immateriali
- II - Immobilizzazioni materiali
  1. terreni e fabbricati
  2. [...]
- [...]

55.5.2 Verifica sull’uso degli elenchi

In base a quanto appreso fino a questo punto, si realizzi il file ‘`verifica-elenchi-01.html`’ che produca un risultato abbastanza simile a quello seguente, quando viene visto attraverso un navigatore:

Passivo

[...]

Fondi per rischi e oneri:

1. fondo per trattamento di quiescenza e obblighi simili;
2. fondo per imposte;
3. altri fondi per rischi e oneri futuri;

Trattamento di fine rapporto di lavoro

Debiti:

1. obbligazioni non convertibili;
2. obbligazioni convertibili;
3. debiti verso banche;
4. [...]

Si completi l’intestazione, coerentemente, secondo il proprio criterio, seguendo le modalità già stabilite. Si deve consegnare per la valutazione:

1. la stampa del risultato ottenuto attraverso il navigatore;
2. la stampa del sorgente.

Si richiede espressamente che tutti gli elementi, a esclusione di quelli che devono essere vuoti, siano terminati correttamente con il marcatore di chiusura.

55.6 Tabelle

La tabella è definita dall’elemento ‘**`TABLE`**’; al suo interno può essere inclusa una didascalia rappresentata dall’elemento ‘**`CAPTION`**’, quindi il contenuto della tabella viene distinto in intestazione, piede e corpo, all’interno dei quali si inseriscono le righe della tabella stessa.

Figura 55.37. Esempio di una tabella.

Codice RGB	Denominazione	
255,0,0	Rosso	intestazione
0,255,0	Verde	corpo
0,0,255	Blu	
Codice RGB	Denominazione	piede

L’intestazione e il piede non sono obbligatori; in ogni caso, se si utilizzano vanno inseriti ordinatamente prima del corpo. Se non si indica l’intestazione o il piede, le righe che costituiscono il corpo vanno delimitate ugualmente tra i marcatori che rappresentano l’elemento corrispondente.

Lo standard ISO 15445 obbliga all’utilizzo dell’attributo ‘**`SUMMARY`**’ nell’elemento ‘**`TABLE`**’. Questo attributo dovrebbe permettere di riassumere il contenuto della tabella per quelle situazioni in cui dovesse essere impossibile consultarla correttamente.

Tabella 55.38. Elementi da usare per la realizzazione delle tabelle HTML.

Elemento	Significato
TABLE	Delimita la tabella.
CAPTION	Didascalia.
THEAD	Righe di intestazione.
TFOOT	Righe del piede.
TBODY	Righe del corpo.
TR	Riga normale (*table row*).
TH	Elemento evidenziato di una riga (*table head*).
TD	Elemento di una riga (*table data*).

L'esempio seguente rappresenta una tabella molto banale, senza intestazione e senza piede:

```
<TABLE SUMMARY="uno due tre quattro cinque sei">
<TBODY>
    <TR><TD><P>uno</P></TD><TD><P>due</P></TD></TR>
    <TR><TD><P>tre</P></TD><TD><P>quattro</P></TD></TR>
    <TR><TD><P>cinque</P></TD><TD><P>sei</P></TD></TR>
</TBODY>
</TABLE>
```

Il risultato è uno specchietto simile a quello che si vede di seguito:

uno	due
tre	quattro
cinque	sei

L'esempio seguente mostra l'aggiunta di una riga di intestazione:

```
<TABLE SUMMARY="uno due tre quattro cinque sei">
<THEAD>
    <TR><TD><P>Primo</P></TD><TD><P>Secondo</P></TD></TR>
</THEAD>
<TBODY>
    <TR><TD><P>uno</P></TD><TD><P>due</P></TD></TR>
    <TR><TD><P>tre</P></TD><TD><P>quattro</P></TD></TR>
    <TR><TD><P>cinque</P></TD><TD><P>sei</P></TD></TR>
</TBODY>
</TABLE>
```

Primo	Secondo
uno	due
tre	quattro
cinque	sei

L'esempio seguente aggiunge anche una didascalia molto breve:

```
<TABLE SUMMARY="uno due tre quattro cinque sei">
<CAPTION>Tabella banale</CAPTION>
<THEAD>
    <TR><TD><P>Primo</P></TD><TD><P>Secondo</P></TD></TR>
</THEAD>
<TBODY>
    <TR><TD><P>uno</P></TD><TD><P>due</P></TD></TR>
    <TR><TD><P>tre</P></TD><TD><P>quattro</P></TD></TR>
    <TR><TD><P>cinque</P></TD><TD><P>sei</P></TD></TR>
</TBODY>
</TABLE>
```

Tabella banale

Primo	Secondo
uno	due
tre	quattro
cinque	sei

Le tabelle HTML possono essere molto più complesse di quanto è stato mostrato qui. Vale la pena di sottolineare il fatto che gli elementi '**TD**', ovvero le celle all'interno delle righe, possono contenere sia testo normale, sia blocchi di testo, ma qui, come negli altri capitoli di queste lezioni, si preferisce usare sempre solo blocchi.

Lo standard ISO 15445 non consente l'utilizzo di attributi per la descrizione dei bordi da far risaltare, perché per questo si possono applicare degli stili, pertanto, inizialmente si ottengono solo tabelle prive di linee di contorno.

55.6.1 File «tabelle-01.html»

Si crei il file '`tabelle-01.html`' con il contenuto seguente:

```
1  <!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
2  <HTML LANG="it">
```



```
3  <HEAD>
4      <META HTTP-EQUIV="Content-Type"
5            CONTENT="text/html; charset=us-ascii">
6      <META NAME="Description"
7            CONTENT="Tabelle con HTML, esempio 01">
8      <META NAME="Keywords"
9            CONTENT="HTML, tabella, tabelle, esempio">
10     <META NAME="Author"
11           CONTENT="Tizio Tizi, tizio@brot.dg, classe 4Z">
12     <META NAME="Date" CONTENT="2012.01.01">
13     <META NAME="Resource-type" LANG="en"
14           CONTENT="Document">
15     <META NAME="Revisit-after" LANG="en"
16           CONTENT="15 days">
17     <META NAME="Robots" CONTENT="ALL">
18     <TITLE>Tabelle</TITLE>
19 </HEAD>
20 <BODY>
21
22 <H1>Operatori</H1>
23
24 <P>Segue una tabella contenente l'elenco degli
25 operatori matematici pi&ugrave; comuni.</P>
26
27 <TABLE SUMMARY="operatori">
28 <CAPTION>Operatori matematici comuni</CAPTION>
29 <THEAD>
30     <TR>
31     <TD><P>Operatore e operandi</P></TD>
32     <TD><P>Operazione</P></TD>
33     <TD><P>Descrizione</P></TD>
34     </TR>
35 </THEAD>
36 <TBODY>
37     <TR>
38     <TD><P><CODE><VAR>op1</VAR>+<VAR>op2</VAR></CODE>
39         </P></TD>
40     <TD><P>somma</P></TD>
41     <TD><P>Somma i due operandi.</P></TD>
42     </TR>
43     <TR>
44     <TD><P><CODE><VAR>op1</VAR>-<VAR>op2</VAR></CODE>
45         </P></TD>
46     <TD><P>sottrazione</P></TD>
47     <TD><P>Sottrae il valore del secondo operando
48         da quello del primo.</P></TD>
49     </TR>
50     <TR>
51     <TD><P><CODE><VAR>op1</VAR>*<VAR>op2</VAR></CODE>
52         </P></TD>
53     <TD><P>moltiplicazione</P></TD>
54     <TD><P>Moltiplica i due operandi.</P></TD>
55     </TR>
56     <TR>
57     <TD><P><CODE><VAR>op1</VAR>/<VAR>op2</VAR></CODE>
58         </P></TD>
59     <TD><P>divisione</P></TD>
60     <TD><P>Divide il primo operando per il valore del
61         secondo.</P></TD>
62     </TR>
63 </TBODY>
64 </TABLE>
65
66 </BODY>
67 </HTML>
```

Nel risultato che si ottiene attraverso un navigatore comune, sono assenti i bordi e non si distinguono le intestazioni dalle altre righe.

Figura 55.46. Aspetto del file ‘tabelle-01.html’ con un navigatore.

Operatori

Segue una tabella contenente l’elenco degli operatori matematici più comuni.

Operatori matematici comuni

Operatore e operandi	Operazione	Descrizione
op1+*op2*	somma	Somma i due operandi.
op1-*op2*	sottrazione	Sottrae il valore del secondo operando da quello del primo.
*op1***op2*	moltiplicazione	Moltiplica i due operandi.
op1/*op2*	divisione	Divide il primo operando per il valore del secondo.

55.6.2 File «tabelle-02.html»

Per esercizio, si salvi il file anche con il nome ‘tabelle-02.html’ e quindi lo si modifichi, in modo da usare l’elemento ‘**TH**’ quando ciò è appropriato, in modo da ottenere un risultato simile a quello seguente:

Operatori

Segue una tabella contenente l’elenco degli operatori matematici più comuni.

Operatori matematici comuni

Operatore e operandi	Operazione	Descrizione
op1+*op2*	somma	Somma i due operandi.
op1-*op2*	sottrazione	Sottrae il valore del secondo operando da quello del primo.
*op1***op2*	moltiplicazione	Moltiplica i due operandi.
op1/*op2*	divisione	Divide il primo operando per il valore del secondo.

55.6.3 Verifica sull’uso di tabelle elementari

In base a quanto appreso fino a questo punto, si realizzi il file ‘verifica-tabelle-01.html’ che produca un risultato abbastanza simile a quello seguente, quando viene visto attraverso un navigatore:

ASCII

Una piccola porzione del codice ASCII.

Ottale	Decimale	Esadecimale	Carattere
141_8	97	61_{16}	a
142_8	98	62_{16}	b
143_8	99	63_{16}	c
144_8	100	64_{16}	d
145_8	101	65_{16}	e
146_8	102	66_{16}	f
147_8	103	67_{16}	g
150_8	104	68_{16}	h
151_8	105	69_{16}	i
152_8	106	$6A_{16}$	j

Si completi l’intestazione, coerentemente, secondo il proprio criterio, seguendo le modalità già stabilite. Si deve consegnare per la valutazione:

1. la stampa del risultato ottenuto attraverso il navigatore;
2. la stampa del sorgente.

Si richiede espressamente che tutti gli elementi, a esclusione di quelli che devono essere vuoti, siano terminati correttamente con il marcatore di chiusura.

55.7 Tabelle complesse

È possibile costruire tabelle un po’ più complesse, utilizzando gli attributi ‘**COLSPAN**’ e ‘**ROWSPAN**’ degli elementi ‘**TD**’ e ‘**TH**’.

Si può immaginare che ogni cella di una tabella possa espandersi orizzontalmente (verso destra) e verticalmente (verso il basso), ma in condizioni normali, ogni cella si espande orizzontalmente e verticalmente di una sola cella.

Quando una cella si espande oltre il proprio spazio, va a occupare quello delle celle adiacenti, che non devono essere dichiarate.

55.7.1 File «tabelle-03.html»

Per esercizio, si riprenda il file ‘tabelle-02.html’ e lo si salvi con il nome ‘tabelle-03.html’, modificandolo nel modo seguente:

```
<!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
<HTML LANG="it">
<HEAD>
    <META HTTP-EQUIV="Content-Type"
          CONTENT="text/html; charset=us-ascii">
    <META NAME="Description" CONTENT="Tabelle complesse
           con HTML, esempio 01">
    <META NAME="Keywords"
          CONTENT="HTML, tabella, tabelle, esempio">
    <META NAME="Author"
          CONTENT="Tizio Tizi, tizio@brot.dg, classe 4Z">
    <META NAME="Date" CONTENT="2012.01.01">
    <META NAME="Resource-type" LANG="en"
          CONTENT="Document">
    <META NAME="Revisit-after" LANG="en"
          CONTENT="15 days">
    <META NAME="Robots" CONTENT="ALL">
    <TITLE>Tabelle</TITLE>
</HEAD>
<BODY>

<H1>Operatori</H1>

<P>Segue una tabella contenente l’elenco degli
operatori matematici pi&ugrave; comuni.</P>

<TABLE SUMMARY="operatori">
<CAPTION>Operatori matematici comuni</CAPTION>
<THEAD>
    <TR>
    <TH></TH>
    <TH ROWSPAN="2"><P>Operatore e operandi</P></TH>
    <TH COLSPAN="2"><P>Annotazioni</P></TH>
    </TR>
    <TR>
    <TH></TH>
    <TH><P>Operazione</P></TH>
    <TH><P>Descrizione</P></TH>
    </TR>
</THEAD>
<TBODY>
    <TR>
    <TH ROWSPAN="4"><P>le quattro operazioni</P></TH>
    <TD><P><CODE><VAR>op1</VAR>+<VAR>op2</VAR></CODE>
        </P></TD>
    <TD><P>somma</P></TD>
    <TD><P>Somma i due operandi.</P></TD>
    </TR>
    <TR>
    <TD><P><CODE><VAR>op1</VAR>-<VAR>op2</VAR></CODE>
        </P></TD>
    <TD><P>sottrazione</P></TD>
    <TD><P>Sottrae il valore del secondo operando
        da quello del primo.</P></TD>
    </TR>
    <TR>
    <TD><P><CODE><VAR>op1</VAR>*<VAR>op2</VAR></CODE>
        </P></TD>
    <TD><P>moltiplicazione</P></TD>
```

```
60      <TD><P>Moltiplica i due operandi.</P></TD>
61      </TR>
62      <TR>
63      <TD><P><CODE><VAR>op1</VAR>/<VAR>op2</VAR></CODE>
64          </P></TD>
65      <TD><P>divisione</P></TD>
66      <TD><P>Divide il primo operando per il valore del
67          secondo.</P></TD>
68      </TR>
69  </TBODY>
70  </TABLE>
71
72  </BODY>
73  </HTML>
```

Nel risultato si può osservare che è stata aggiunta una colonna, che nel corpo si mostra come una cella unica, usata come intestazione delle righe; inoltre, nell'intestazione superiore la cella contenente la stringa «Operatore e operandi» si espande in basso occupando lo spazio della cella inferiore, mentre la cella contenente la stringa «Annotazioni» occupa anche lo spazio della cella successiva a destra.

Figura 55.50. Aspetto del file '`tabelle-03.html`'. Per facilitare l'individuazione delle aree occupate dalle celle, è stato aggiunto un bordo attorno a ogni cella.

Operatori

Segue una tabella contenente l'elenco degli operatori matematici più comuni.

Operatori matematici comuni

	Operatore e operandi	Annotazioni	
		Operazione	Descrizione
le quattro operazioni	*op1*+*op2*	somma	Somma i due operandi.
	op1−*op2*	sottrazione	Sottrae il valore del secondo operando da quello del primo.
	*op1***op2*	moltiplicazione	Moltiplica i due operandi.
	op1/*op2*	divisione	Divide il primo operando per il valore del secondo.

55.7.2 Verifica sull'uso di tabelle complesse

«

In base a quanto appreso fino a questo punto, si realizzi il file '`verifica-tabelle-complesse-01.html`' che produca un risultato abbastanza simile a quello seguente, quando viene visto attraverso un navigatore. Per facilitare l'interpretazione dell'immagine, sono stati inseriti i bordi attorno alla tabella, ma la verifica richiede la produzione di una tabella normale, senza bordi:

ASCII

Una piccola porzione del codice ASCII.

Codifica			Carattere
Ottale	Decimale	Esadecimale	
141_8	97	61_{16}	a
142_8	98	62_{16}	b
143_8	99	63_{16}	c
144_8	100	64_{16}	d
145_8	101	65_{16}	e
146_8	102	66_{16}	f
147_8	103	67_{16}	g
150_8	104	68_{16}	h
151_8	105	69_{16}	i
152_8	106	$6A_{16}$	j

Si completi l'intestazione, coerentemente, secondo il proprio criterio, seguendo le modalità già stabilite. Si deve consegnare per la valutazione:

1. la stampa del risultato ottenuto attraverso il navigatore;
2. la stampa del sorgente.

Si richiede espressamente che tutti gli elementi, a esclusione di quelli che devono essere vuoti, siano terminati correttamente con il marcatore di chiusura.

55.8 Riferimenti ipertestuali

«

La sigla HTML fa riferimento esplicitamente a un sistema ipertestuale (*Hypertext markup language*), pertanto, la realizzazione di collegamenti del genere è una funzionalità essenziale di questo tipo di documento.

Un riferimento ipertestuale può essere fatto a una pagina intera o a un punto particolare di una pagina. Il riferimento può essere assoluto, cioè provvisto dell'indicazione del nodo e del percorso necessario a raggiungere la pagina, oppure può essere relativo al nodo attuale, oppure anche relativo al percorso stesso della pagina da cui parte il riferimento.

Per i riferimenti ipertestuali si utilizza l'elemento '**`A`**' ed eventualmente l'attributo '**`ID`**' di molti altri elementi.

Un riferimento a una pagina intera, con l'indicazione del percorso assoluto per raggiungerla, viene fatto come nell'esempio seguente:

```
<A HREF="http://www.brot.dg/prove/prova.html">Pagina
 di prova</A>
```

Nell'esempio, la frase «Pagina di prova» serve come riferimento a '`http://www.brot.dg/prove/prova.html`'.

Quando di realizza un documento HTML composto da più pagine collegate tra loro, è preferibile utilizzare riferimenti relativi, in modo da non dover indicare il nome del nodo in cui si trovano e nemmeno il percorso assoluto delle directory da attraversare per raggiungerle.

```
<A HREF="varie/nota.html">Annotazioni varie</A>
```

Nell'esempio, si vede un riferimento al file '`nota.html`' contenuto nella «directory» '`varie/`' discendente dalla directory corrente. La directory corrente, in questi casi, è quella in cui si trova la pagina contenente il puntatore.

All'interno di una pagina è possibile collocare delle etichette che poi possono servire per fare dei riferimenti, sia a partire dalla stessa

pagina, sia da altre. L'esempio seguente mostra una situazione molto semplice:

```
<A NAME="introduzione"></A>
```

Si usa quindi lo stesso elemento che serve per creare un puntatore, ma con l'attributo '**NAME**'. L'argomento dell'attributo '**NAME**' (in questo caso è la parola '**introduzione**') identifica quel punto.

Per fare riferimento a un'etichetta nella stessa pagina si può usare la forma dell'esempio seguente, con il quale si vuole puntare all'etichetta appena creata.

```
<A HREF="#introduzione">Introduzione</A>
```

Si utilizza l'attributo '**HREF**' come al solito, ma il suo argomento è il nome dell'etichetta preceduta dal simbolo '#'. Evidentemente, ciò è necessario per evitare di fare riferimento a un file con lo stesso nome.

Se si vuole fare riferimento a un'etichetta di un certo file, si utilizza la notazione solita, aggiungendo l'indicazione dell'etichetta:

```
<A HREF="http://www.brot.dg/varie/linux.html#introduzione">
Introduzione a GNU/Linux</A>
```

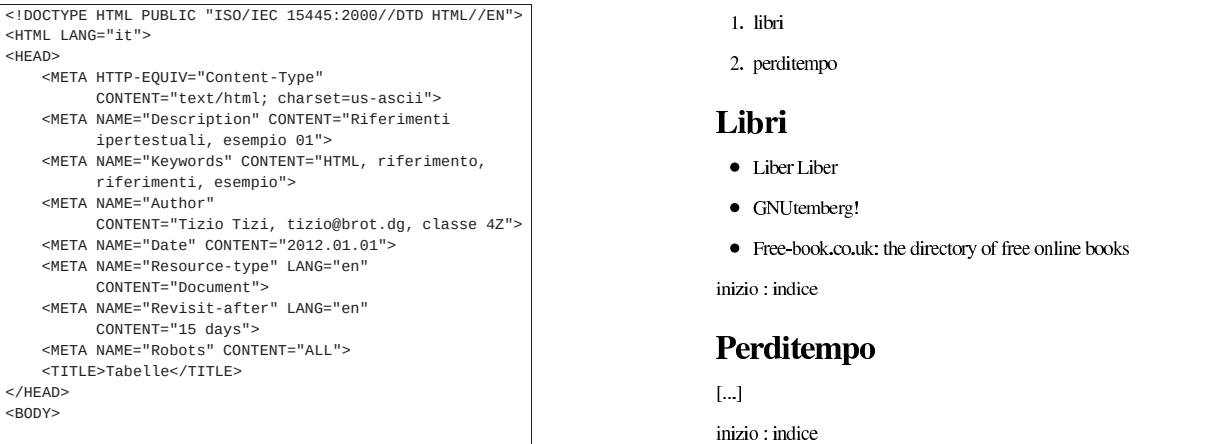
55.8.1 File «riferimenti-01.html»

Si realizzi il file '`riferimenti-01.html`' con il contenuto seguente:

```
1  <!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
2  <HTML LANG="it">
3  <HEAD>
4      <META HTTP-EQUIV="Content-Type"
5            CONTENT="text/html; charset=us-ascii">
6      <META NAME="Description" CONTENT="Riferimenti
7            ipertestuali, esempio 01">
8      <META NAME="Keywords" CONTENT="HTML, riferimento,
9            riferimenti, esempio">
10     <META NAME="Author"
11           CONTENT="Tizio Tizi, tizio@brot.dg, classe 4Z">
12     <META NAME="Date" CONTENT="2012.01.01">
13     <META NAME="Resource-type" LANG="en"
14           CONTENT="Document">
15     <META NAME="Revisit-after" LANG="en"
16           CONTENT="15 days">
17     <META NAME="Robots" CONTENT="ALL">
18     <TITLE>Tabelle</TITLE>
19 </HEAD>
20 <BODY>
21
22 <H1><A NAME="indice">Indice</A></H1>
23
24 <OL>
25 <LI>
26
27     <P><A HREF="#libri">libri</A></P>
28
29 </LI>
30 <LI>
31
32     <P><A HREF="#perditempo">perditempo</A></P>
33
34 </LI>
35 </OL>
36
37 <H1><A NAME="libri">Libri</A></H1>
38
39 <UL>
40 <LI>
41
42     <P><A HREF="http://www.liberliber.it">Liber
43     Liber</A></P>
44
45 </LI>
46 <LI>
47
48     <P><A HREF="http://www.gnutemberg.org">
49     GNUtemberg!</A></P>
50
51 </LI>
52 <LI>
53
54     <P><A HREF="http://www.free-book.co.uk">
55     Free-book.co.uk: the directory of free online
56     books</A></P>
57
58 </LI>
59 </UL>
60
61 <P><A HREF="riferimenti-01.html">inizio</A> : <A
62 HREF="#indice">indice</A></P>
63
64 <H1><A NAME="perditempo">Perditempo</A></H1>
65
66 <P>[...]</P>
67
68 <P><A HREF="riferimenti-01.html">inizio</A> : <A
69 HREF="#indice">indice</A></P>
70
71 </BODY>
72 </HTML>
```


Come si può osservare, sono presenti riferimenti interni al testo e riferimenti esterni. I riferimenti esterni, sono assoluti e devono specificare anche il protocollo di comunicazione (*http://...*); oltre ai riferimenti interni fatti a delle etichette, sono presenti dei riferimenti relativi allo stesso file (a titolo di esempio).

Figura 55.58. Aspetto del file '`riferimenti-01.html`'.



Indice

1. libri
2. perditempo

Libri

- Liber Liber
- GNUtemberg!
- Free-book.co.uk: the directory of free online books

inizio : indice

Perditempo

[...]

inizio : indice

55.8.2 Verifica sull'uso dei riferimenti ipertestuali

In base a quanto appreso fino a questo punto, si realizzi un sistema simile a quello del file '`riferimenti-01.html`', composto dai file: '`verifica-riferimenti-01-indice.html`', '`verifica-riferimenti-01-libri.html`', '`verifica-riferimenti-01-perditempo.html`'. I contenuti di questi file devono riportare rispettivamente l'indice, l'elenco dei libri e del «perditempo», come già avviene nel file '`riferimenti-01.html`', ma in modo separato. I riferimenti tra questi file devono essere relativi; si osservi che la voce «inizio» deve portare all'inizio dei file stessi e non alla pagina dell'indice.

Si deve consegnare per la valutazione:

1. la stampa del risultato ottenuto attraverso il navigatore dei tre file;
2. la stampa del sorgente dei tre file.

Si richiede espressamente che tutti gli elementi, a esclusione di quelli che devono essere vuoti, siano terminati correttamente con il marcatore di chiusura.

55.9 Citazioni

Per la scrittura di una citazione, sono previsti due elementi: '**BLOCKQUOTE**' e '**Q**'. Il primo dei due, come suggerisce il nome, è un blocco e può contenere blocchi, mentre il secondo si inserisce nel testo lineare e può contenere solo testo lineare.

I due elementi '**BLOCKQUOTE**' e '**Q**' sono accomunati dalla disponibilità dell'attributo '**CITE**', con il quale si può specificare un indirizzo URI dove trovare il documento originale.

55.9.1 File «citazioni-01.html»

Si realizzi il file '`citazioni-01.html`' con il contenuto seguente:

```
<!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
<HTML LANG="it">
<HEAD>
    <META HTTP-EQUIV="Content-Type"
          CONTENT="text/html; charset=us-ascii">
    <META NAME="Description"
          CONTENT="Citazioni, esempio 01">
    <META NAME="Keywords"
          CONTENT="HTML, citazione, citazioni, esempio">
    <META NAME="Author"
          CONTENT="Tizio Tizi, tizio@brot.dg, classe 4Z">
    <META NAME="Date" CONTENT="2012.01.01">
    <META NAME="Resource-type" LANG="en"
          CONTENT="Document">
    <META NAME="Revisit-after" LANG="en"
          CONTENT="15 days">
    <META NAME="Robots" CONTENT="ALL">
    <TITLE>Citazioni</TITLE>
</HEAD>
<BODY>

<H1>Licenza GNU GPL</H1>

<P>La licenza GNU GPL, all'articolo tre riporta in
particolare quanto segue:</P>

<BLOCKQUOTE CITE="http://www.gnu.org/licenses/gpl.html">

    <P>3. You may copy and distribute the Program (or a
    work based on it, under Section 2) in object code
    or executable form under the terms of Sections 1
    and 2 above provided that you also do one of the
    following:</P>

    <P>[<EM>omissis</EM>]</P>

    <P>c) Accompany it with the information you
    received as to the offer to distribute
    corresponding source code. (This alternative is
    allowed only for noncommercial distribution and
    only if you received the program in object code or
    executable form with such an offer, in accord with
    Subsection b above.)</P>

</BLOCKQUOTE>

<P>Subito dopo, compare la frase: <Q
CITE="http://www.gnu.org/licenses/gpl.html">The source
code for a work means the preferred form of the work
for making modifications to it</Q>.</P>

</BODY>
</HTML>
```

Nella figura successiva si vede il risultato che si dovrebbe ottenere attraverso un navigatore. Se il navigatore è abbastanza evoluto (per esempio si può provare con Amaya), dalle citazioni si arriva al documento originale presso *http://www.gnu.org/licenses/gpl.html* (con Amaya è sufficiente fare un clic doppio quando il puntatore del mouse è sulla superficie del testo relativo).

Figura 55.60. Aspetto del file '`citazioni-01.html`'.

Licenza GNU GPL

La licenza GNU GPL, all'articolo tre riporta in particolare quanto segue:

> 3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:
>
> [*omissis*]
>
> c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

Subito dopo, compare la frase: «The source code for a work means the preferred form of the work for making modifications to it».

55.10 Inserzione di oggetti

Un documento HTML può contenere riferimenti a componenti esterni, che però devono concorrere alla realizzazione del documento finale, così come viene reso dal navigatore. Nei casi più comuni si tratta di immagini o di applicazioni grafiche (spesso si tratta di applicazioni Java o Flash), ma il concetto riguarda qualunque altra cosa che possa essere incorporata nel documento. L'elemento attraverso cui si includono gli oggetti è '**OBJECT**'. La tabella successiva elenca alcuni degli attributi di questo elemento.

Tabella 55.61. Alcuni attributi dell'elemento '**OBJECT**'.

Attributo	Significato
DATA	Riferimento al file dell'oggetto.
TYPE	Tipo di oggetto.
STANDBY	Messaggio di attesa durante il caricamento dell'oggetto.

L'attributo '**DATA**' è indispensabile; inoltre, in generale è opportuno aggiungere anche l'attributo '**TYPE**' per precisare subito il tipo di oggetto.

L'elemento '**OBJECT**' non può essere vuoto; ciò che racchiude è quanto deve essere mostrato nel caso non sia possibile raggiungere l'oggetto indicato, oppure non sia possibile gestire l'oggetto stesso. Di solito si tratta di testo normale, ma potrebbe trattarsi di altri oggetti alternativi.

Nel caso particolare dell'incorporazione di immagini, per l'esigenza di mantenere la compatibilità con il passato, rimane disponibile l'elemento '**IMG**', che richiede l'uso dell'attributo '**SRC**' per specificare la posizione del file da visualizzare e permette di usare l'attributo '**ALT**' per annotare un testo da mostrare in mancanza della possibilità di visualizzare l'immagine. A differenza dell'elemento '**OBJECT**', l'elemento '**IMG**' è vuoto.

Sia l'elemento '**OBJECT**', sia l'elemento '**IMG**' si utilizzano in un contesto lineare.

Gli oggetti che si inseriscono in un documento HTML devono essere visualizzati, o comunque interpretati dal navigatore. Quando si realizzano documenti (anche pubblicitari), che devono essere rivolti alla maggiore quantità di utenti possibili, occorre valutare la diffusione del formato degli oggetti che si vogliono inserire, per evitare di escludere gli utenti che non dispongono del software necessario. A titolo di esempio, si consideri che gli interpreti Java e Flash, necessari a eseguire le applicazioni rispettive, non sono distribuiti come software libero, pertanto, chi utilizza soltanto software libero non può visualizzare tali applicazioni (o quantomeno non può farlo in modo completo).

55.10.1 File «oggetti-01.html»

Si realizzi il file '`oggetti-01.html`' con il contenuto seguente. Per completare il lavoro, serve il file '`philosophical-gnu-sm.jpg`', che si può prelevare da *http://www.gnu.org/graphics/philosophical-*

gnu-sm.jpg . Il disegno è di Markus Gerwinski e il titolo è *Philosophical GNU*. In mancanza di un collegamento alla rete, si può usare un'immagine qualunque, purché in formato JPG che è quello più comune, modificando in modo appropriato i riferimenti al nome del file relativo.

```
<!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
<HTML LANG="it">
<HEAD>
    <META HTTP-EQUIV="Content-Type"
          CONTENT="text/html; charset=us-ascii">
    <META NAME="Description"
          CONTENT="Incorporazione di oggetti,
                   esempio 01">
    <META NAME="Keywords"
          CONTENT="HTML, oggetto, oggetti, esempio">
    <META NAME="Author"
          CONTENT="Tizio Tizi, tizio@brot.dg, classe 4Z">
    <META NAME="Date" CONTENT="2012.01.01">
    <META NAME="Resource-type" LANG="en"
          CONTENT="Document">
    <META NAME="Revisit-after" LANG="en"
          CONTENT="15 days">
    <META NAME="Robots" CONTENT="ALL">
    <TITLE>Oggetti</TITLE>
</HEAD>
<BODY>

<H1>Inclusione di oggetti</H1>

<P>Per includere un oggetto si usa l'elemento
<SAMP>OBJECT</SAMP>, come in questo caso, in cui si
incorpora un file JPG nel flusso del testo di questo
paragrafo: <OBJECT DATA="philosophical-gnu-sm.jpg"
TYPE="image/jpg">GNU filosofico</OBJECT>. L'oggetto,
oltre che disporre di un testo alternativo, potrebbe
incorporare anche un oggetto di riserva, come
nell'esempio seguente, dove il primo riferimento
&egrave; all'immagine originale, ma in sua mancanza si
mostra un file che accompagna il documento stesso
(oppure si ripiega ancora per un testo
descrittivo):</P>

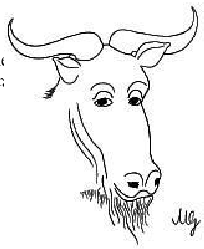
<P><OBJECT
DATA="http://www.gnu.org/graphics/philosophical-gnu-sm.jpg"
          TYPE="image/jpg">
    <OBJECT DATA="philosophical-gnu-sm.jpg"
            TYPE="image/jpg">
        GNU filosofico
    </OBJECT>
</OBJECT></P>

<P>Infine, per essere certi che tutto funzioni anche
con un vecchio navigatore, &egrave; sempre meglio
limitarsi a usare l'elemento <SAMP>IMG</SAMP> per le
immagini: <IMG SRC="philosophical-gnu-sm.jpg" ALT="GNU
filosofico">.</P>

</BODY>
</HTML>
```

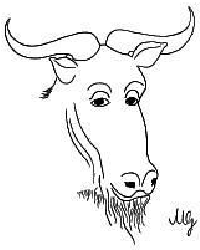
Ciò che si ottiene cambia molto anche in base all'ampiezza orizzontale disponibile, dal momento che la prima e l'ultima immagine sono inserite nel testo di un paragrafo (solo la seconda immagine è inserita in un paragrafo autonomo che non contiene altro testo).

Figura 55.63. Aspetto del file '`oggetti-01.html`'.



Inclusione di oggetti

Per includere un oggetto si usa l'elemento OBJECT, come in questo caso, in cui si incorpora un file JPG nel flusso del testo di questo paragrafo:  . L'oggetto, oltre che disporre di un testo alternativo, potrebbe incorporare anche un oggetto di riserva, come nell'esempio seguente, dove il primo riferimento è all'immagine originale, ma in sua mancanza si mostra un file che accompagna il documento stesso (oppure si ripiega ancora per un testo descrittivo):



Infine, per essere certi che tutto funzioni anche con un vecchio navigatore, è sempre meglio limitarsi a usare l'elemento IMG per le immagini: .

Viene mostrato anche cosa si vede con un navigatore non grafico:

```
                    Inclusione di oggetti

  Per includere un oggetto si usa l'elemento OBJECT, come in
  questo caso, in cui si incorpora un file JPG nel flusso del
  testo di questo paragrafo: GNU filosofico. L'oggetto, oltre
  che disporre di un testo alternativo, potrebbe incorporare
  anche un oggetto di riserva, come nell'esempio seguente,
  dove il primo riferimento è all'immagine originale, ma in
  sua mancanza si mostra un file che accompagna il documento
  stesso (oppure si ripiega ancora per un testo descrittivo):

  GNU filosofico

  Infine, per essere certi che tutto funzioni anche con un
  vecchio navigatore, è sempre meglio limitarsi a usare
  l'elemento IMG per le immagini: GNU filosofico.
```

Generalmente, per evitare problemi di compatibilità con i vari programmi di navigazione, è meglio evitare di fare scorrere il testo a fianco delle immagini, per cui è bene staccare il testo normale racchiudendolo esplicitamente all'interno di un elemento '**P**' (paragrafo).

55.10.2 File «oggetti-02.html»

Si realizzi il file '`oggetti-02.html`' con il contenuto seguente:

```
<!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
<HTML LANG="it">
<HEAD>
    <META HTTP-EQUIV="Content-Type"
          CONTENT="text/html; charset=us-ascii">
    <META NAME="Description"
          CONTENT="Incorporazione di oggetti,
                   esempio 02">
    <META NAME="Keywords"
          CONTENT="HTML, oggetto, oggetti, esempio">
    <META NAME="Author"
          CONTENT="Tizio Tizi, tizio@brot.dg, classe 4Z">
    <META NAME="Date" CONTENT="2012.01.01">
    <META NAME="Resource-type" LANG="en"
          CONTENT="Document">
    <META NAME="Revisit-after" LANG="en"
          CONTENT="15 days">
    <META NAME="Robots" CONTENT="ALL">
    <TITLE>Convalida</TITLE>
```

```
</HEAD>
<BODY>

<H1>Fieri di essere standard</H1>

<P>Quando si &egrave; certi della validit&agrave; del
proprio documento HTML ISO 15445, si pu&ograve;
aggiungere un'icona racchiusa da un riferimento
ipertestuale che rimanda al servizio di verifica
offerto dal consorzio W3C. Per verificare che questo
documento sia valido effettivamente, basta fare un
&laquo;clic&raquo; sulla stessa:</P>

<P><A HREF="http://validator.w3.org/check/referer"><IMG
SRC="http://validator.w3.org/images/v15445" ALT="Valid
ISO-HTML!"></A></P>

<P>Se per&ograve; questo file non &egrave; accessibile
attraverso la rete esterna, ci si deve accontentare di
verificare a partire dalla pagina <A
HREF="http://validator.w3.org/file-upload.html">
http://validator.w3.org/file-upload.html</A>,
attraverso la quale &egrave; possibile inviare il file,
pur non essendo accessibile pubblicamente.</P>

</BODY>
</HTML>
```

Il risultato che si ottiene con un navigatore grafico lo si può vedere nella figura successiva. Si osservi che l'icona può apparire solo se è disponibile un collegamento alla rete esterna, che consenta di raggiungere l'indirizzo *http://validator.w3.org/images/v15445* .

Figura 55.66. Aspetto del file '`oggetti-02.html`'.

Fieri di essere standard

Quando si è certi della validità del proprio documento HTML ISO 15445, si può aggiungere un'icona racchiusa da un riferimento ipertestuale che rimanda al servizio di verifica offerto dal consorzio W3C. Per verificare che questo documento sia valido effettivamente, basta fare un «clic» sulla stessa:

Se però questo file non è accessibile attraverso la rete esterna, ci si deve accontentare di verificare a partire dalla pagina http://validator.w3.org/file-upload.html, attraverso la quale è possibile inviare il file, pur non essendo accessibile pubblicamente.

Ecco cosa si vede con un navigatore senza grafica:

```
                    Fieri di essere standard

  Quando si è certi della validità del proprio documento
  HTML ISO 15445, si può aggiungere un'icona racchiusa da un
  riferimento ipertestuale che rimanda al servizio di verifica
  offerto dal consorzio W3C. Per verificare che questo
  documento sia valido effettivamente, basta fare un «clic»
  sulla stessa:

  Valid ISO-HTML!

  Se però questo file non è accessibile attraverso la rete
  esterna, ci si deve accontentare di verificare a partire
  dalla pagina http://validator.w3.org/file-upload.html,
  attraverso la quale è possibile inviare il file, pur non
  essendo accessibile pubblicamente.
```

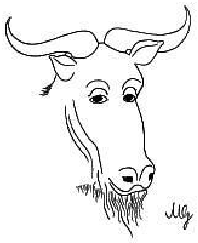
55.10.3 Verifica sull'inserzione di oggetti

«

In base a quanto appreso fino a questo punto, si realizzi il file '`verifica-oggetti-01.html`' che produca un risultato abbastanza simile a quello mostrato di seguito (sia in modo grafico, sia in modo non grafico), quando viene visto attraverso un navigatore:

Le cose che mi piacciono

Ci sono due cose che mi piacciono: il software libero e gli standard liberi. Per saperne di più sul software libero si può leggere qualcosa di utile a partire dal sito http://www.gnu.org (basta fare clic sull'immagine del "GNU filosofico" che appare qui sotto).



Per quanto riguarda gli standard, si veda in particolare il W3C (http://www.w3.org), ma per verificare la correttezza sintattica di questa pagina basta fare un bel clic sulla superficie di questa icona:

```
                    Le cose che mi piacciono

  Ci sono due cose che mi piacciono: il software libero e gli
  standard liberi. Per saperne di più sul software libero si
  può leggere qualcosa di utile a partire dal sito
  http://www.gnu.org (basta fare clic sull'immagine del "GNU
  filosofico" che appare qui sotto).

  GNU filosofico

  Per quanto riguarda gli standard, si veda in particolare il
  W3C (http://www.w3.org), ma per verificare la correttezza
  sintattica di questa pagina basta fare un bel clic sulla
  superficie di questa icona: Valid ISO-HTML!
```

Si completi l'intestazione, coerentemente, secondo il proprio criterio, seguendo le modalità già stabilite. Inoltre, le figure devono incorporare il riferimento ipertestuale appropriato al contesto. Si deve consegnare per la valutazione:

1. la stampa del risultato ottenuto attraverso il navigatore;
2. la stampa del sorgente.

Si richiede espressamente che tutti gli elementi, a esclusione di quelli che devono essere vuoti, siano terminati correttamente con il marcatore di chiusura.

55.11 Inserzione di immagini

«

L'elemento '**`OBJECT`**' consente di incorporare qualunque tipo di «oggetto», incluse le immagini; tuttavia, ci sono dei navigatori che considerano tale elemento in modo speciale e pretendono che sia disponibile del software specifico per qualunque tipo di contenuto multimediale. In altri termini, dovendo realizzare un documento da consultare con qualunque tipo di navigatore, l'inserzione delle immagini conviene attuarla con l'elemento tradizionale '**`IMG`**' ed è opportuno che i file siano in formato JPG.

Tabella 55.70. Attributi indispensabili dell'elemento '**`IMG`**'.

Attributo	Significato
SRC	Riferimento al file dell'immagine.
ALT	Descrizione alternativa all'immagine.

L'elemento '**`IMG`**' può essere vuoto, perché si limita a stabilire la collocazione dell'immagine all'interno di un contesto lineare.

55.11.1 File «immagini-01.html»

«

Si realizzi il file '`immagini-01.html`' con il contenuto seguente. Per completare il lavoro, serve il file '`philosophical-gnu-sm.jpg`', che si può prelevare da *http://www.gnu.org/graphics/philosophical-gnu-sm.jpg* . Il disegno è di Markus Gerwinski e il titolo è *Philosophical GNU*. In mancanza di un collegamento alla rete,

si può usare un'immagine qualunque, purché in formato JPG che è quello più comune, modificando in modo appropriato i riferimenti al nome del file relativo.

```
<!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
<HTML LANG="it">
<HEAD>
    <META HTTP-EQUIV="Content-Type"
          CONTENT="text/html; charset=us-ascii">
    <META NAME="Description"
          CONTENT="Incorporazione di immagini,
                   esempio 01">
    <META NAME="Keywords"
          CONTENT="HTML, immagine, immagini, esempio">
    <META NAME="Author"
          CONTENT="Tizio Tizi, tizio@brot.dg, classe 4Z">
    <META NAME="Date" CONTENT="2012.01.01">
    <META NAME="Resource-type" LANG="en"
          CONTENT="Document">
    <META NAME="Revisit-after" LANG="en"
          CONTENT="15 days">
    <META NAME="Robots" CONTENT="ALL">
    <TITLE>Immagini</TITLE>
</HEAD>
<BODY>

<H1>Inclusione di immagini</H1>

<P>Per includere un'immagine si usa l'elemento
<SAMP>IMG</SAMP>, come in questo caso, in cui si
incorpora un file JPG nel flusso del testo di questo
paragrafo: <IMG SRC="philosophical-gnu-sm.jpg" ALT="GNU
filosofico">.</P>

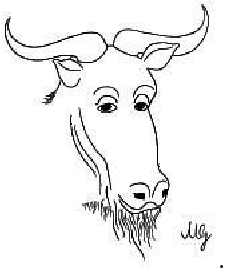
</BODY>
</HTML>
```

Ciò che si ottiene cambia anche in base all'ampiezza orizzontale disponibile.

Figura 55.72. Aspetto del file '`immagini-01.html`'.

Inclusione di immagini

Per includere un'immagine si usa l'elemento `IMG`, come in questo caso, in cui si incorpora un file JPG nel flusso del testo di questo paragrafo:



Viene mostrato anche cosa si vede con un navigatore non grafico:

```
                    Inclusione di immagini

Per includere un'immagine si usa l'elemento IMG, come in
questo caso, in cui si incorpora un file JPG nel flusso del
testo di questo paragrafo: GNU filosofico.
```

Generalmente, per evitare problemi di compatibilità con i vari programmi di navigazione, è meglio evitare di fare scorrere il testo a fianco delle immagini, per cui è bene staccare il testo normale racchiudendolo esplicitamente all'interno di un elemento '**P**' (paragrafo).

55.11.2 File «immagini-02.html»

Si realizzi il file '`immagini-02.html`' con il contenuto seguente:

```
<!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
<HTML LANG="it">
<HEAD>
    <META HTTP-EQUIV="Content-Type"
          CONTENT="text/html; charset=us-ascii">
    <META NAME="Description"
          CONTENT="Incorporazione di immagini,
                   esempio 02">
    <META NAME="Keywords"
          CONTENT="HTML, immagine, immagini, esempio">
    <META NAME="Author"
          CONTENT="Tizio Tizi, tizio@brot.dg, classe 4Z">
    <META NAME="Date" CONTENT="2012.01.01">
    <META NAME="Resource-type" LANG="en"
          CONTENT="Document">
    <META NAME="Revisit-after" LANG="en"
          CONTENT="15 days">
    <META NAME="Robots" CONTENT="ALL">
    <TITLE>Convalida</TITLE>
</HEAD>
<BODY>

<H1>Fieri di essere standard</H1>

<P>Quando si &egrave; certi della validit&agrave; del
proprio documento HTML ISO 15445, si pu&ograve;
aggiungere un'icona racchiusa da un riferimento
ipertestuale che rimanda al servizio di verifica
offerto dal consorzio W3C. Per verificare che questo
documento sia valido effettivamente, basta fare un
&laquo;clic&raquo; sulla stessa:</P>

<P><A HREF="http://validator.w3.org/check/referer"><IMG
SRC="http://validator.w3.org/images/v15445" ALT="Valid
ISO-HTML!"></A></P>

<P>Se per&ograve; questo file non &egrave; accessibile
attraverso la rete esterna, ci si deve accontentare di
verificare a partire dalla pagina <A
HREF="http://validator.w3.org/file-upload.html">
http://validator.w3.org/file-upload.html</A>,
attraverso la quale &egrave; possibile inviare il file,
pur non essendo accessibile pubblicamente.</P>

</BODY>
</HTML>
```


Il risultato che si ottiene con un navigatore grafico lo si può vedere nella figura successiva. Si osservi che l'icona può apparire solo se è disponibile un collegamento alla rete esterna, che consenta di raggiungere l'indirizzo *http://validator.w3.org/images/v15445* .

Figura 55.75. Aspetto del file '`oggetti-02.html`'.

Fieri di essere standard

Quando si è certi della validità del proprio documento HTML ISO 15445, si può aggiungere un'icona racchiusa da un riferimento ipertestuale che rimanda al servizio di verifica offerto dal consorzio W3C. Per verificare che questo documento sia valido effettivamente, basta fare un «clic» sulla stessa:

Se però questo file non è accessibile attraverso la rete esterna, ci si deve accontentare di verificare a partire dalla pagina http://validator.w3.org/file-upload.html, attraverso la quale è possibile inviare il file, pur non essendo accessibile pubblicamente.

Ecco cosa si vede con un navigatore senza grafica:

```
                    Fieri di essere standard

Quando si è certi della validità del proprio documento
HTML ISO 15445, si può aggiungere un'icona racchiusa da un
riferimento ipertestuale che rimanda al servizio di verifica
offerto dal consorzio W3C. Per verificare che questo
documento sia valido effettivamente, basta fare un «clic»
sulla stessa:

Valid ISO-HTML!

Se però questo file non è accessibile attraverso la rete
esterna, ci si deve accontentare di verificare a partire
dalla pagina http://validator.w3.org/file-upload.html,
attraverso la quale è possibile inviare il file, pur non
essendo accessibile pubblicamente.
```

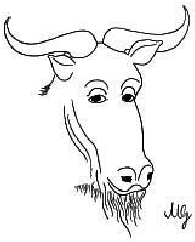
55.11.3 Verifica sull'inserzione di immagini

Questa verifica è identica a quella della sezione 55.10.3.

In base a quanto appreso fino a questo punto, si realizzi il file '`verifica-immagini-01.html`' che produca un risultato abbastanza simile a quello mostrato di seguito (sia in modo grafico, sia in modo non grafico), quando viene visto attraverso un navigatore:

Le cose che mi piacciono

Ci sono due cose che mi piacciono: il software libero e gli standard liberi. Per saperne di più sul software libero si può leggere qualcosa di utile a partire dal sito http://www.gnu.org (basta fare clic sull'immagine del "GNU filosofico" che appare qui sotto).



Per quanto riguarda gli standard, si veda in particolare il W3C (http://www.w3.org), ma per verificare la correttezza sintattica di questa pagina basta fare un bel clic sulla superficie di questa icona: 


```
                Le cose che mi piacciono

  Ci sono due cose che mi piacciono: il software libero e gli
  standard liberi. Per saperne di più sul software libero si
  può leggere qualcosa di utile a partire dal sito
  http://www.gnu.org (basta fare clic sull'immagine del "GNU
  filosofico" che appare qui sotto).

  GNU filosofico

  Per quanto riguarda gli standard, si veda in particolare il
  W3C (http://www.w3.org), ma per verificare la correttezza
  sintattica di questa pagina basta fare un bel clic sulla
  superficie di questa icona: Valid ISO-HTML!
```

Si completi l'intestazione, coerentemente, secondo il proprio criterio, seguendo le modalità già stabilite. Inoltre, le figure devono incorporare il riferimento ipertestuale appropriato al contesto. Si deve consegnare per la valutazione:

1. la stampa del risultato ottenuto attraverso il navigatore;
2. la stampa del sorgente.

Si richiede espressamente che tutti gli elementi, a esclusione di quelli che devono essere vuoti, siano terminati correttamente con il marcatore di chiusura.

55.12 Fogli di stile CSS

L'evoluzione del linguaggio HTML si è sviluppata attraverso la ricerca di introdurre elementi e attributi per il controllo della resa estetica. Tuttavia, questo approccio ha dimostrato di non essere efficace e lo standard ISO 15445 ha eliminato tutto il superfluo, per lasciare il controllo dell'estetica al di fuori del linguaggio, contando eventualmente sui fogli di stile CSS per questo scopo.

Un foglio di stile CSS (*Cascading style sheet*) può essere un file, di solito con estensione '`.css`', che si associa alle pagine HTML, oppure può essere del codice che si incorpora nelle pagine stesse. Si associa un foglio di stile esterno nel modo che appare dall'esempio seguente, dove il file del foglio di stile si chiama precisamente '`stile.css`':


```
<!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
<HTML>
<HEAD>
    <TITLE>Esempio</TITLE>
    <LINK REL="stylesheet" TYPE="text/css" HREF="stile.css">
    ...
</HEAD>
...
</HTML>
```

Per incorporare il contenuto di un foglio di stile direttamente nella pagina HTML, si può procedere come nell'esempio seguente:

```
<!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
<HTML>
<HEAD>
    <TITLE>Esempio</TITLE>
    <STYLE TYPE="text/css">
        <!--
            H1 { color: blue }
            P {
                font-size:  12pt;
                color:      red;
            }
        -->
    </STYLE>
</HEAD>
<BODY>
    ...
</BODY>
</HTML>
```

Come si può osservare, il codice dello stile CSS è inserito nell'elemento '**STYLE**', ma appare racchiuso da un commento (tra '`<!--`' e '`-->`'). Ciò si rende necessario per evitare che i navigatori che non sono in grado di interpretare lo stile vengano confusi, arrivando magari a mostrare il codice CSS nella pagina.

Se si dispone di un collegamento alla rete esterna, è possibile verificare la correttezza sintattica di un foglio di stile, attraverso il servizio offerto dal consorzio W3C, a partire dall'indirizzo *http://jigsaw.w3.org/css-validator/* . La verifica è possibile per file HTML che incorporano il foglio di stile, per figli di stile autonomi, o inserendo il testo dello stile in una finestra di un modulo di inserimento.

55.12.1 File «stile-01.html»

Si realizzi il file '`stile-01.html`' con il contenuto seguente:

```
 1  <!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
 2  <HTML LANG="it">
 3  <HEAD>
 4      <META HTTP-EQUIV="Content-Type"
 5            CONTENT="text/html; charset=us-ascii">
 6      <META NAME="Description"
 7            CONTENT="Introduzione all'uso degli
 8                     stili CSS, esempio 01">
 9      <META NAME="Keywords"
10            CONTENT="HTML, CSS, stile, esempio">
11      <META NAME="Author"
12            CONTENT="Tizio Tizi, tizio@brot.dg, classe 4Z">
13      <META NAME="Date" CONTENT="2012.01.01">
14      <META NAME="Resource-type" LANG="en"
15            CONTENT="Document">
16      <META NAME="Revisit-after" LANG="en"
17            CONTENT="15 days">
18      <META NAME="Robots" CONTENT="ALL">
19      <TITLE>Stile</TITLE>
20      <STYLE TYPE="text/css">
21          <!--
22              H1 {
23                  color: blue;
24                  font-size:  32pt;
25              }
26              P {
27                  font-size:  24pt;
28                  color:      red;
29              }
30          -->
31      </STYLE>
32  </HEAD>
33  <BODY>
34
```

```
<H1>Fieri di essere standard</H1>

<P>Quando si &egrave; certi della validit&agrave; del
proprio documento HTML ISO 15445, si pu&ograve;
aggiungere un'icona racchiusa da un riferimento
ipertestuale che rimanda al servizio di verifica
offerto dal consorzio W3C. Per verificare che questo
documento sia valido effettivamente, basta fare un
&laquo;clic&raquo; sulla stessa:</P>

<P><A HREF="http://validator.w3.org/check/referer"><IMG
SRC="http://validator.w3.org/images/v15445" ALT="Valid
ISO-HTML!"></A></P>

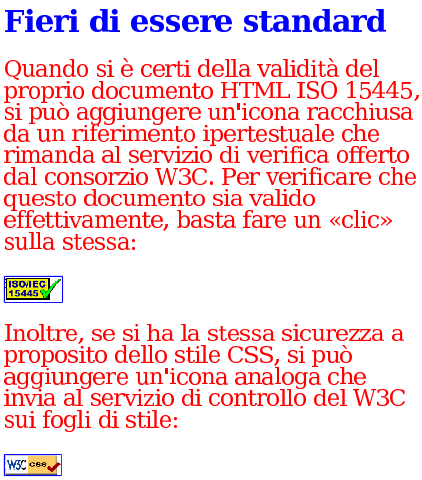
<P>Inoltre, se si ha la stessa sicurezza a proposito
dello stile CSS, si pu&ograve; aggiungere un'icona
analoga che invia al servizio di controllo del W3C sui
fogli di stile:</P>

<P><A
HREF="http://jigsaw.w3.org/css-validator/check/validator">
<IMG SRC="http://jigsaw.w3.org/css-validator/images/vcss"
ALT="Valid CSS!"></A></P>

</BODY>
</HTML>
```

Teoricamente, il titolo dovrebbe apparire di colore blu e il testo di colore rosso, come si vede nella figura successiva.

Figura 55.82. Aspetto del file '`stile-01.html`'.



55.12.2 File «stile-02.html» e «stile-02.css»

Si realizzi il file '`stile-02.css`' estrapolando il codice del foglio di stile CSS dal file '`stile-01.html`'. In pratica, il file '`stile-02.css`' deve avere il contenuto seguente:

```
H1 {
    color: blue;
    font-size:  32pt;
}
P {
    font-size:  24pt;
    color:      red;
}
```

Si realizzi il file '`stile-02.html`', modificando il file '`stile-01.html`' già realizzato in precedenza. In pratica, si fa riferimento al foglio di stile contenuto nel file '`stile-02.css`':

```
<!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
<HTML LANG="it">
<HEAD>
    <META HTTP-EQUIV="Content-Type"
          CONTENT="text/html; charset=us-ascii">
    <META NAME="Description"
          CONTENT="Introduzione all'uso degli
                   stili CSS, esempio 02">
    <META NAME="Keywords"
          CONTENT="HTML, CSS, stile, esempio">
```



```
    <META NAME="Author"
          CONTENT="Tizio Tizi, tizio@brot.dg, classe 4Z">
    <META NAME="Date" CONTENT="2012.01.01">
    <META NAME="Resource-type" LANG="en"
          CONTENT="Document">
    <META NAME="Revisit-after" LANG="en"
          CONTENT="15 days">
    <META NAME="Robots" CONTENT="ALL">
    <TITLE>Stile</TITLE>
    <LINK REL="stylesheet" TYPE="text/css"
          HREF="stile-02.css">
</HEAD>
<BODY>

<H1>Fieri di essere standard</H1>

<P>Quando si &egrave; certi della validit&agrave; del
proprio documento HTML ISO 15445, si pu&ograve;
aggiungere un'icona racchiusa da un riferimento
ipertestuale che rimanda al servizio di verifica
offerto dal consorzio W3C. Per verificare che questo
documento sia valido effettivamente, basta fare un
&laquo;clic&raquo; sulla stessa:</P>

<P><A HREF="http://validator.w3.org/check/referer"><IMG
SRC="http://validator.w3.org/images/v15445" ALT="Valid
ISO-HTML!"></A></P>

<P>Inoltre, se si ha la stessa sicurezza a proposito
dello stile CSS, si pu&ograve; aggiungere un'icona
analoga che invia al servizio di controllo del W3C sui
fogli di stile:</P>

<P><A
HREF="http://jigsaw.w3.org/css-validator/check/validator">
<IMG SRC="http://jigsaw.w3.org/css-validator/images/vcss"
ALT="Valid CSS!"></A></P>

</BODY>
</HTML>
```

Si deve ottenere lo stesso risultato già visto nella sezione precedente, a proposito del file '`stile-01.html`'.

55.13 Linguaggio CSS

In queste lezioni viene introdotto un utilizzo molto semplice dei fogli di stile CSS (si veda anche la sezione 54.6). Esistono due livelli del linguaggio CSS, denominati CSS1 e CSS2, compatibili tra di loro. Eventualmente si può approfondire lo studio di questi due livelli da *http://www.w3.org/TR/REC-CSS1* e da *http://www.w3.org/TR/CSS2/*.

Le istruzioni del linguaggio CSS vengono definite ***regole*** e si esprimono sinteticamente secondo la forma seguente, dove le parentesi graffe fanno parte della regola:

selettore { ***dichiarazione*** }

Il ***selettore*** rappresenta qualcosa all'interno del documento e la dichiarazione è ciò che si vuole ottenere su tale oggetto. Eventualmente, all'interno di una regola si possono raggruppare più selettori (separati con una virgola), applicando così le stesse dichiarazioni:

selettore , ***selettore*** ... { ***dichiarazione*** }

Si possono indicare anche più dichiarazioni, separandole con un punto e virgola:

selettore ... { ***dichiarazione*** ; ***dichiarazione*** ... }

Le regole possono essere scritte utilizzando più righe di testo normale, per cui, la stessa sintassi appena mostrata potrebbe essere scritta anche come nel modo seguente, aggiungendo un punto e virgola finale anche dopo l'ultima dichiarazione:

```
selettore... {
    dichiarazione ;
    dichiarazione ;
    ...
}
```

Le dichiarazioni si scompongono a loro volta in ***proprietà*** e valori loro assegnati:

```
selettore... {
    proprietà : valore ;
    ...
}
```

L'esempio seguente attribuisce il colore blu al testo degli elementi '**H1**' di un documento HTML, come già apparso in precedenza:

```
H1 { color: blue }
```

Una proprietà può avere valori alternativi, da usare in mancanza di altro:

```
selettore... {
    proprietà : valore valore_alternativo... ;
    ...
}
```

L'esempio successivo indica l'utilizzo di uno sfondo composto da un'immagine esterna per il corpo del documento, specificando che in mancanza dell'immagine, o in mancanza della possibilità di rappresentarla si può utilizzare uno sfondo bianco:

```
BODY { background-image: url(fondale.jpg) white }
```

I commenti in un foglio di stile CSS si rappresentano in modo simile al linguaggio C, nella forma:

```
/* testo_ignorato */
```

Il selettore di una regola CSS è qualcosa che rappresenta una parte del testo a cui si vogliono applicare le dichiarazioni relative. Nella situazione più semplice, il selettore viene indicato con il nome dell'elemento a cui si attribuisce, come già apparso negli esempi mostrati in precedenza. Tutti gli elementi HTML che si possono utilizzare nel corpo, ovvero all'interno dell'elemento '**BODY**', possono utilizzare l'attributo '**CLASS**'. Ciò permette di attribuire loro una ***classe***, ovvero un gruppo, di solito nell'ambito di quel tipo di elemento. Per indicare un selettore che faccia riferimento a una classe specifica di un certo elemento, si usa la notazione seguente:

```
[elemento] . classe
```

Come si vede, l'indicazione dell'elemento è facoltativa, in modo tale che, se non lo si indica, si faccia riferimento a tutti gli elementi che appartengono a quella stessa classe. L'esempio seguente mostra il caso degli elementi '**P**' che appartengono alla classe '**nota**', a cui viene abbinato il colore rosso per il testo:

```
P.nota { color: red }
```

L'esempio seguente mostra invece l'utilizzo di un selettore che fa riferimento a una classe di qualunque elemento:

```
.calmante { color: green }
```

Un selettore può essere composto in modo da definire la dipendenza da un contesto. In altri termini, si può definire un selettore che dipende da un altro:

```
selettore sottoselettore...
```

Il primo selettore indica un ambito, all'interno del quale va cercata la corrispondenza per il secondo selettore, continuando eventualmente ad aumentare il dettaglio con altri selettori più specifici. Si osservi l'esempio seguente; serve a fare riferimento agli elementi '**EM**' che si trovano all'interno di un elemento '**H1**':

```
H1 EM { color: green }
```

È importante distinguere il raggruppamento di selettori dalla definizione di un contesto più dettagliato come in questo caso. Infatti, per raggruppare i selettori si utilizza la virgola. L'esempio seguente applica il colore verde a tutti gli elementi '**EM**' contenuti all'interno di elementi '**H1**' o '**H2**':

```
H1 EM, H2 EM { color: green }
```

Un selettore può anche individuare una pseudo-classe, ovvero una zona di testo che viene individuata dal programma che si occupa di interpretare il documento HTML, che non corrisponde a elementi e classi indicati espressamente:

```
elemento : pseudo_classe
```

```
elemento . classe : pseudo_classe
```

```
. classe : pseudo_classe
```

```
: pseudo_classe
```

Il caso tipico di una pseudo-classe è quella che delimita la prima lettera di un elemento: '**first-letter**'. L'esempio seguente serve a ottenere una lettera iniziale più grande in tutti gli elementi '**P**' di classe '**primo**':

```
P.primo:first-letter {
    font-size: 200%;
    float: left;
}
```

55.13.1 File «linguaggio-css-01.html»

Si realizzi il file '`linguaggio-css-01.html`' con il contenuto seguente:

```
<!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
<HTML LANG="it">
<HEAD>
    <META HTTP-EQUIV="Content-Type"
          CONTENT="text/html; charset=us-ascii">
    <META NAME="Description"
          CONTENT="Linguaggio CSS, esempio 01">
    <META NAME="Keywords"
          CONTENT="HTML, CSS, stile, esempio">
    <META NAME="Author"
          CONTENT="Tizio Tizi, tizio@brot.dg, classe 4Z">
    <META NAME="Date" CONTENT="2012.01.01">
    <META NAME="Resource-type" LANG="en"
          CONTENT="Document">
    <META NAME="Revisit-after" LANG="en"
          CONTENT="15 days">
    <META NAME="Robots" CONTENT="ALL">
    <TITLE>Linguaggio CSS</TITLE>
    <STYLE TYPE="text/css">
        <!--
            H1 {
                color: blue;
                font-size:        48pt;
            }
            P {
                font-size:        24pt;
                color:            green;
            }
            P.autore {
                font-size:        18pt;
                text-align:       right;
                font-style:       italic;
                color:            black;
            }
            P.inizio:first-letter {
                font-size:        200%;
                color:            black;
            }
```

```
        -->
    </STYLE>
</HEAD>
<BODY>

<H1>H&auml;nsel e Gretel</H1>

<P CLASS="autore">di Jacob e Wilhelm Grimm</P>

<P CLASS="inizio">Davanti a un gran bosco abitava un
povero taglialegna con sua moglie e i suoi due bambini;
il maschietto si chiamava H&auml;nsel e la bambina
Gretel. Egli aveva poco da metter sotto i denti, e
quando ci fu nel paese una grande carestia, non poteva
neanche pi&ugrave; procurarsi il pane tutti i giorni.
[...]</P>

<P>Per la fame, neppure i due bimbi potevan dormire, e
avevano udito quel che la matrigna diceva al padre.
Gretel piangeva amaramente, e disse a H&auml;nsel: -
Adesso per noi
&egrave; finita. - [...]</P>

<P>[...]</P>

</BODY>
</HTML>
```

Figura 55.93. Aspetto del file '`linguaggio-css-01.html`'.

Hänsel e Gretel

di Jacob e Wilhelm Grimm

Davanti a un gran bosco abitava un povero taglialegna con sua moglie e i suoi due bambini; il maschietto si chiamava Hänsel e la bambina Gretel. Egli aveva poco da metter sotto i denti, e quando ci fu nel paese una grande carestia, non poteva neanche più procurarsi il pane tutti i giorni. [...]

Per la fame, neppure i due bimbi potevan dormire, e avevano udito quel che la matrigna diceva al padre. Gretel piangeva amaramente, e disse a Hänsel: – Adesso per noi è finita. – [...]

[...]

55.13.2 Elementi «DIV» e «SPAN»

Nelle sezioni di questo capitolo che descrivono l'uso del linguaggio HTML è stata omessa la descrizione degli elementi '**`DIV`**' e '**`SPAN`**', i quali diventano importanti quando si usano gli stili CSS. L'elemento '**`DIV`**' è un blocco che racchiude blocchi; l'elemento '**`SPAN`**' si inserisce nel testo lineare e contiene testo lineare.

Questi elementi non comportano una modifica estetica del loro contenuto, perché servono solo a delimitare in qualche modo blocchi o testo, per assegnargli degli attributi. Per esempio, si può modificare il linguaggio:

```
<DIV LANG="en">

    <P>This is free software; 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>

</DIV>
```

Nell'ambito dello studio dei fogli di stile, questi elementi sono utili per l'attribuzione di una classe:

```
Il sig. <SPAN CLASS="nome">Tizio Tizi</SPAN> è andato...
```


55.14 Controllo del carattere

Il carattere tipografico può essere controllato con una serie di proprietà CSS; in particolare possono essere utili quelle che appaiono nella tabella successiva.

Tabella 55.96. Proprietà riferite ai caratteri.

Proprietà	Valori	Descrizione
`font-family`	***nome***	Carattere tipografico indicato per nome.
	`serif`	Carattere con grazie.
	`sans-serif`	Carattere senza grazie.
	`monospace`	Dattilografico.
`font-style`	`normal`	Forma normale.
	`italic`	Corsivo.
	`oblique`	Obliquo.
`font-variant`	`normal`	Serie normale.
	`small-caps`	Maiuscoletto.
`font-weight`	`normal`	Tono normale.
	`bold`	Nero.
	`bolder`	Nerissimo.
	`lighter`	Chiaro.
`font-size`	***n*`pt`	Dimensione in punti.
	***n*`cm`	Dimensione in centimetri.
	***n*`mm`	Dimensione in millimetri.
	***n*`em`	Dimensione relativa in quadratoni.
	***n*`ex`	Dimensione relativa in Ex.
	***n*`%`	Dimensione relativa percentuale.
	`small`	Carattere piccolo.
	`medium`	Carattere normale.
	`large`	Carattere grande.

Alla proprietà '**`font-family`**' può essere attribuito il nome di una famiglia di caratteri, oppure il nome di una «famiglia generica», che in pratica identifica uno stile del carattere senza indicare esattamente quale tipo di carattere (nella tabella appaiono solo nomi di famiglie generiche). Una famiglia di caratteri potrebbe essere '**`times`**', mentre una famiglia generica potrebbe essere '**`serif`**', ovvero un carattere munito di grazie. Alla proprietà '**`font-family`**' possono essere abbinati più tipi di caratteri, separati da una virgola, per indicare una sequenza alternativa da utilizzare in mancanza di quello preferito:

```
BODY { font-family: gill, helvetica, sans-serif }
```

L'esempio mostra proprio questo: prima si tenta di utilizzare il carattere '**`gill`**'; quindi si prova con '**`helvetica`**'; infine ci si accontenta di un carattere senza grazie, '**`sans-serif`**'.

Figura 55.98. Confronto visivo tra le famiglie generiche.

Times serif carattere con grazie

Helvetica sans–serif carattere senza grazie

Courier monospace carattere dattilografico

55.14.1 File «caratteri-css-01.html»

Si realizzi il file ‘`caratteri-css-01.html`’ partendo dal file ‘`linguaggio-css-01.html`’, già realizzato in precedenza, modificando la dichiarazione dello stile come si vede nel testo seguente:

```
<!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
<HTML LANG="it">
<HEAD>
    <META HTTP-EQUIV="Content-Type"
          CONTENT="text/html; charset=us-ascii">
    <META NAME="Description"
          CONTENT="Carattere tipografico, esempio 01">
    <META NAME="Keywords"
          CONTENT="HTML, CSS, carattere, esempio">
    <META NAME="Author"
          CONTENT="Tizio Tizi, tizio@brot.dg, classe 4Z">
    <META NAME="Date" CONTENT="2012.01.01">
    <META NAME="Resource-type" LANG="en"
          CONTENT="Document">
    <META NAME="Revisit-after" LANG="en"
          CONTENT="15 days">
    <META NAME="Robots" CONTENT="ALL">
    <TITLE>Linguaggio CSS</TITLE>
    <STYLE TYPE="text/css">
        <!--
            H1 {
                font-family:    serif;
                font-style:     normal;
                font-variant:   normal;
                font-weight:    bolder;
                font-size:      2cm;
            }
            P {
                font-family:    sans-serif;
                font-style:     italic;
                font-variant:   normal;
                font-weight:    lighter;
                font-size:      1cm;
            }
            P.autore {
                font-size:        5mm;
                text-align:       right;
                font-style:       italic;
            }
            P.inizio:first-letter {
                font-size:        200%;
                color:            Blue;
            }
        -->
    </STYLE>
</HEAD>
<BODY>

<H1>H&auml;nsel e Gretel</H1>

<P CLASS="autore">di Jacob e Wilhelm Grimm</P>

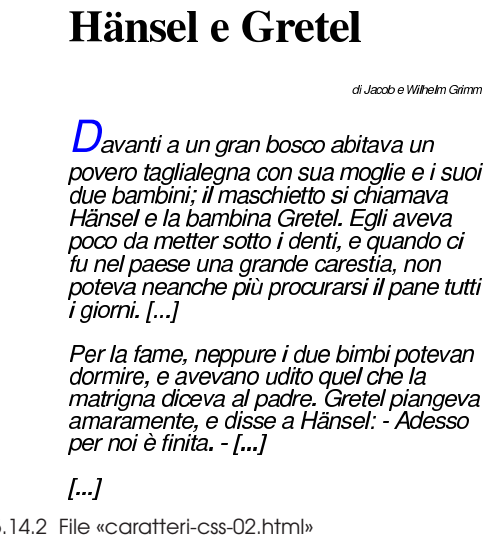
<P CLASS="inizio">Davanti a un gran bosco abitava un
povero taglialegna con sua moglie e i suoi due bambini;
il maschietto si chiamava H&auml;nsel e la bambina
Gretel. Egli aveva poco da metter sotto i denti, e
quando ci fu nel paese una grande carestia, non poteva
neanche pi&ugrave; procurarsi il pane tutti i giorni.
[...]</P>

<P>Per la fame, neppure i due bimbi potevan dormire, e
avevano udito quel che la matrigna diceva al padre.
Gretel piangeva amaramente, e disse a H&auml;nsel: -
Adesso per noi
&egrave; finita. - [...]</P>

<P>[...]</P>

</BODY>
</HTML>
```


Figura 55.100. Aspetto del file ‘`caratteri-css-01.html`’.



55.14.2 File «caratteri-css-02.html»

Si realizzi il file ‘`caratteri-css-02.html`’, con qualsiasi contenuto, sperimentando anche altre proprietà descritte nel capitolo a proposito del carattere tipografico.

55.14.3 Verifica sull’uso delle proprietà riferite ai caratteri

In base a quanto appreso fino a questo punto, si realizzi il file ‘`verifica-caratteri-css-01.html`’ che produca un risultato abbastanza simile a quello mostrato di seguito, quando viene visto attraverso un navigatore. In questa verifica va usato in modo appropriato l’elemento ‘**SPAN**’:

Le sillabe della prima riga e le lettere della seconda, rappresentano le sette note musicali, scritte secondo la notazione italiana e la notazione inglese. Ogni nota (sillaba o lettera) è scritta con un rapporto di circa il 150 % rispetto alla precedente, inoltre sono stati usati gli evidenziamenti seguenti:

Nota (sillaba o lettera)	Carattere
do	Carattere con grazie, normale.
re	Carattere con grazie, corsivo.
mi	Carattere senza grazie, normale.
fa	Carattere senza grazie, corsivo.
sol	Carattere dattilografico, normale.
la	Carattere dattilografico, corsivo
si	Carattere con grazie, maiuscoletto.
c	Carattere con grazie, normale, nero.
d	Carattere con grazie, corsivo, nero.
e	Carattere senza grazie, normale, nero.
f	Carattere senza grazie, corsivo, nero.
g	Carattere dattilografico, normale, nero.
a	Carattere dattilografico, corsivo, nero.
b	Carattere con grazie, maiuscoletto, nero.

Si completi l’intestazione, coerentemente, secondo il proprio cri-

terio, seguendo le modalità già stabilite. Inoltre, lo stile CSS deve essere incorporato nel file HTML. Si deve consegnare per la valutazione:

1. la stampa del risultato ottenuto attraverso il navigatore;
2. la stampa del sorgente.

Si richiede espressamente che tutti gli elementi, a esclusione di quelli che devono essere vuoti, siano terminati correttamente con il marcatore di chiusura.

55.15 Colore e sfondo

Il colore del carattere, il colore dello sfondo ed eventualmente l'immagine dello sfondo possono essere definiti attraverso le proprietà CSS della tabella successiva.

Tabella 55.103. Proprietà riferite ai colori e allo sfondo.

Proprietà	Valori	Descrizione
`color`	***colore***	Colore del carattere o di primo piano.
`background-color`	***colore***	Colore dello sfondo.
`background-image`	`url(`***uri***`)`	Immagine da usare per lo sfondo.

I colori si possono indicare attraverso il nome che questi hanno in inglese, oppure attraverso la funzione ***rgb()***, con la quale si specifica il valore RGB:

`rgb(`***livello_rosso***`,` ***livello_verde***`,` ***livello_blu***`)`

I numeri che esprimono i livelli dei colori fondamentali RGB vanno da 0 a 255.

Quando si esprimono i colori attraverso il nome (in inglese), conviene rimanere nell'ambito di un gruppo ristretto: '**`aqua`**', '**`black`**', '**`blue`**', '**`fuchsia`**', '**`gray`**', '**`green`**', '**`lime`**', '**`maroon`**', '**`navy`**', '**`olive`**', '**`purple`**', '**`red`**', '**`silver`**', '**`teal`**', '**`white`**' e '**`yellow`**'.

55.15.1 File «colori-css-01.html»

Si realizzi il file '`colori-css-01.html`' con il contenuto seguente, partendo da una copia del file '`caratteri-css-01.html`':

```
<!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
<HTML LANG="it">
<HEAD>
    <META HTTP-EQUIV="Content-Type"
          CONTENT="text/html; charset=us-ascii">
    <META NAME="Description"
          CONTENT="Colori del testo e dello sfondo,
                   esempio 01">
    <META NAME="Keywords"
          CONTENT="HTML, CSS, carattere, esempio">
    <META NAME="Author"
          CONTENT="Tizio Tizi, tizio@brot.dg, classe 4Z">
    <META NAME="Date" CONTENT="2012.01.01">
    <META NAME="Resource-type" LANG="en"
          CONTENT="Document">
    <META NAME="Revisit-after" LANG="en"
          CONTENT="15 days">
    <META NAME="Robots" CONTENT="ALL">
    <TITLE>Linguaggio CSS</TITLE>
    <STYLE TYPE="text/css">
        <!--
            BODY {
                background-color: yellow;
            }
            H1 {
                font-weight:      bolder;
                font-size:        2cm;
                background-color: red;
                color:            white;
            }
            P {
                font-family:      sans-serif;
                font-style:       italic;
                font-variant:     normal;
                font-weight:      lighter;
                font-size:        1cm;
                background-color: green;
            }
            P.autore {
                font-size:        5mm;
                text-align:       right;
                font-style:       italic;
                color:            white;
            }
            P.inizio:first-letter {
                font-size:        200%;
                color:            Blue;
            }
        -->
    </STYLE>
</HEAD>
<BODY>

<H1>H&auml;nsel e Gretel</H1>

<P CLASS="autore">di Jacob e Wilhelm Grimm</P>

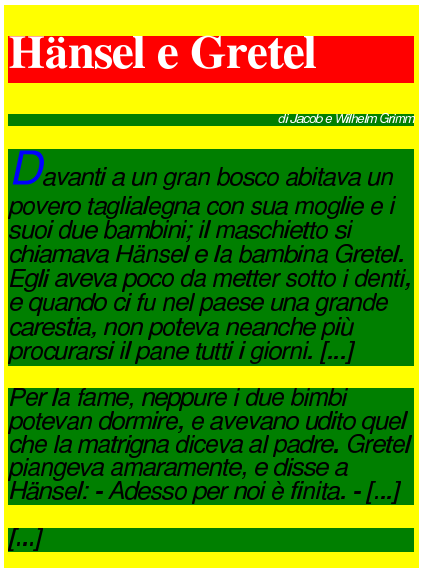
<P CLASS="inizio">Davanti a un gran bosco abitava un
povero taglialegna con sua moglie e i suoi due bambini;
il maschietto si chiamava H&auml;nsel e la bambina
Gretel. Egli aveva poco da metter sotto i denti, e
quando ci fu nel paese una grande carestia, non poteva
neanche pi&ugrave; procurarsi il pane tutti i giorni.
[...]</P>

<P>Per la fame, neppure i due bimbi potevan dormire, e
avevano udito quel che la matrigna diceva al padre.
Gretel piangeva amaramente, e disse a H&auml;nsel: -
Adesso per noi
&egrave; finita. - [...]</P>

<P>[...]</P>

</BODY>
</HTML>
```

Figura 55.105. Aspetto del file '`colori-css-01.html`'.



55.15.2 File «colori-css-02.html»

Si realizzi il file '`colori-css-02.html`' con un contenuto qualsiasi, sperimentando l'uso di altri colori comuni: '**`aqua`**', '**`black`**', '**`blue`**', '**`fuchsia`**', '**`gray`**', '**`green`**', '**`lime`**', '**`maroon`**', '**`navy`**', '**`olive`**', '**`purple`**', '**`red`**', '**`silver`**', '**`teal`**', '**`white`**' e '**`yellow`**'.

55.15.3 File «sfondo-css-01.html»

Si realizzi il file ‘`sfondo-css-01.html`’ con il contenuto seguente:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
    "http://www.w3.org/TR/html4/strict.dtd">
<HTML LANG="it">
<HEAD>
    <META HTTP-EQUIV="Content-Type"
          CONTENT="text/html; charset=us-ascii">
    <META NAME="Description" CONTENT="Sfondo">
    <META NAME="Keywords"
          CONTENT="HTML, sfondo, esempio 1">
    <META NAME="Author"
          CONTENT="Tizio Tizi, tizio@brot.dg, classe 4Z">
    <META NAME="Date" CONTENT="2012.01.01">
    <META NAME="Resource-type" LANG="en"
          CONTENT="Document">
    <META NAME="Revisit-after" LANG="en"
          CONTENT="15 days">
    <META NAME="Robots" CONTENT="ALL">
    <TITLE>Sfondo</TITLE>
    <STYLE TYPE="text/css">
        <!--
            BODY {
                background-color: yellow;
                background-image:
url(http://www.gnu.org/graphics/philosophical-gnu-sm.jpg);
            }
            H1 {
                font-weight:    bolder;
                font-size:      2cm;
            }
            P {
                font-family:    sans-serif;
                font-size:      1cm;
            }
        -->
    </STYLE>
</HEAD>
<BODY>

<H1>Le cose che mi piacciono</H1>

<P>Ci sono due cose che mi piacciono: il software
libero e gli standard liberi. Per saperne di pi&ugrave;
sul software libero si pu&ograve; leggere qualcosa di
utile a partire dal sito <A
HREF="http://www.gnu.org">http://www.gnu.org</A>.</P>

</BODY>
</HTML>
```

Si deve ottenere sullo sfondo del documento l’immagine del «GNU filosofico», ripetuta più volte (se non si dispone di un collegamento alla rete esterna, si può usare un’immagine qualunque, modificando il riferimento in modo appropriato, per esempio così: ‘**`url(philosophical-gnu-sm.jpg)`**’).

55.15.4 File «sfondo-css-02.html»

Si realizzi il file ‘`sfondo-css-02.html`’ a partire dal file ‘`sfondo-css-01.html`’, facendo in modo che l’immagine compaia solo sotto la superficie dei paragrafi (gli elementi ‘**`P`**’).

55.15.5 Verifica sull’uso delle proprietà riferite ai colori

In base a quanto appreso fino a questo punto, si realizzi il file ‘`verifica-colori-css-01.html`’ che produca un risultato abbastanza simile a quello mostrato di seguito, quando viene visto attraverso un navigatore. Per realizzare questa verifica si può partire dal file ‘`verifica-caratteri-css-01.html`’ già realizzato in precedenza:

Lo sfondo del documento è grigio; ogni nota musicale è rappresentata con un colore diverso, con questa sequenza: nero, marrone, rosso, arancio, giallo, verde, blu.

Si completi l’intestazione, coerentemente, secondo il proprio criterio, seguendo le modalità già stabilite. Inoltre, lo stile CSS deve essere incorporato nel file HTML. Si deve consegnare per la valutazione:

1. la stampa del risultato ottenuto attraverso il navigatore;
2. la stampa del sorgente.

Si richiede espressamente che tutti gli elementi, a esclusione di quelli che devono essere vuoti, siano terminati correttamente con il marcatore di chiusura.

55.16 Testo

Alcune proprietà riferite al testo sono descritte nella tabella successiva. Si tratta in particolare dell’allineamento orizzontale, del rientro e dell’altezza della riga.

Tabella 55.108. Proprietà riferite al testo.

Proprietà	Valori	Descrizione
`vertical-align`	`baseline`	Testo al livello normale.
	`middle`	Allinea al centro.
	`sub`	Pedice.
	`super`	Apice.
`text-transform`	`none`	Nessuna trasformazione del testo.
	`capitalize`	Rende maiuscola la prima lettera delle parole.
	`uppercase`	Tutto maiuscolo.
	`lowercase`	Tutto minuscolo.
`text-align`	`left`	Allinea a sinistra.
	`right`	Allinea a destra.
	`center`	Centra.
	`justify`	Allinea a sinistra e a destra.
`text-indent`	*n*`pt`	Rientro in punti.
	n`cm`	Rientro in centimetri.
	n`mm`	Rientro in millimetri.
	n`em`	Rientro relativo in quadratoni.
	n`ex`	Rientro relativo in Ex.

Proprietà	Valori	Descrizione
	n%	Rientro relativo in percentuale.
line-height	normal	Altezza normale della riga.
	*n*pt	Altezza in punti.
	*n*cm	Altezza in centimetri.
	*n*mm	Altezza in millimetri.
	n%	Altezza relativa in percentuale.

55.16.1 File «testo-css-01.html»

Si realizzi il file ‘testo-css-01.html’ con il contenuto seguente, partendo da una copia del file ‘colori-css-01.html’:

```
<!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
<HTML LANG="it">
<HEAD>
    <META HTTP-EQUIV="Content-Type"
          CONTENT="text/html; charset=us-ascii">
    <META NAME="Description"
          CONTENT="Caratteristiche del testo,
                   esempio 01">
    <META NAME="Keywords"
          CONTENT="HTML, CSS, testo, esempio">
    <META NAME="Author"
          CONTENT="Tizio Tizi, tizio@brot.dg, classe 4Z">
    <META NAME="Date" CONTENT="2012.01.01">
    <META NAME="Resource-type" LANG="en"
          CONTENT="Document">
    <META NAME="Revisit-after" LANG="en"
          CONTENT="15 days">
    <META NAME="Robots" CONTENT="ALL">
    <TITLE>Linguaggio CSS</TITLE>
    <STYLE TYPE="text/css">
        <!--
            BODY {
                line-height:      15mm;
            }
            H1 {
                font-size:        10mm;
                text-transform:   uppercase;
            }
            P {
                font-size:        5mm;
                text-align:       justify;
            }
        -->
    </STYLE>
</HEAD>
<BODY>

<H1>H&auml;nsel e Gretel</H1>

<P>di Jacob e Wilhelm Grimm</P>

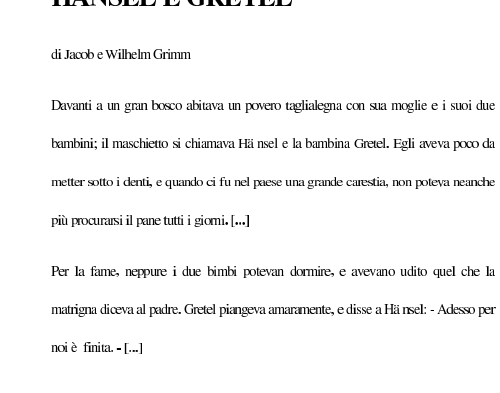
<P>Davanti a un gran bosco abitava un povero
taglialegna con sua moglie e i suoi due bambini; il
maschietto si chiamava H&auml;nsel e la bambina Gretel.
Egli aveva poco da metter sotto i denti, e quando ci fu
nel paese una grande carestia, non poteva neanche
pi&ugrave; procurarsi il pane tutti i giorni. [...]</P>

<P>Per la fame, neppure i due bimbi potevan dormire, e
avevano udito quel che la matrigna diceva al padre.
Gretel piangeva amaramente, e disse a H&auml;nsel: -
Adesso per noi
&egrave; finita. - [...]</P>

<P>[...]</P>

</BODY>
</HTML>
```

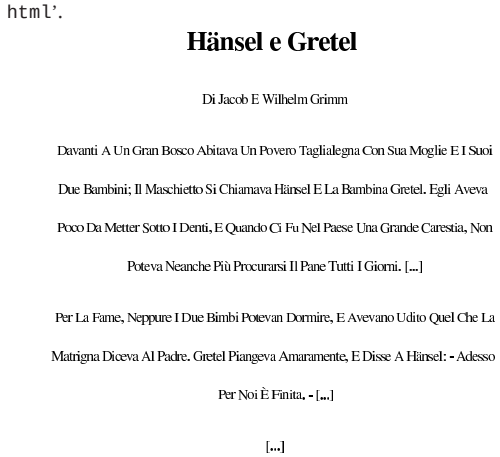

Figura 55.110. Aspetto del file ‘testo-css-01.html’.



55.16.2 Verifica sull’uso delle proprietà riferite al testo

In base a quanto appreso fino a questo punto, si realizzi il file ‘verifica-testo-css-01.html’ che produca un risultato abbastanza simile a quello mostrato di seguito, quando viene visto attraverso un navigatore:

Figura 55.111. Aspetto del file ‘verifica-testo-css-01.html’.



Le maiuscole all’inizio di ogni parola sono ottenute attraverso una proprietà, mentre nel sorgente HTML il testo è scritto normalmente.

Si completi l’intestazione, coerentemente, secondo il proprio criterio, seguendo le modalità già stabilite. Inoltre, lo stile CSS deve essere incorporato nel file HTML. Si deve consegnare per la valutazione:

1. la stampa del risultato ottenuto attraverso il navigatore;
2. la stampa del sorgente.

Si richiede espressamente che tutti gli elementi, a esclusione di quelli che devono essere vuoti, siano terminati correttamente con il marcatore di chiusura.

55.17 Blocchi

Alcune proprietà riferite ai blocchi di testo rettangolari sono descritte nella tabella successiva. Si tratta in particolare dei margini, dei bordi e della dichiarazione di blocchi fluttuanti.

Tabella 55.112. Proprietà riferite al testo racchiuso in blocchi rettangolari.

Proprietà	Valori	Descrizione
margin-top	auto	Margine superiore automatico.
	*n*pt	Margine superiore in punti.

Proprietà	Valori	Descrizione
	*n*cm	Margine superiore in centimetri.
	*n*mm	Margine superiore in millimetri.
	n%	Margine superiore relativo in percentuale.
margin-bottom	auto	Margine inferiore automatico.
	*n*pt	Margine inferiore in punti.
	*n*cm	Margine inferiore in centimetri.
	*n*mm	Margine inferiore in millimetri.
	n%	Margine inferiore relativo in percentuale.
margin-left	auto	Margine sinistro automatico.
	*n*pt	Margine sinistro in punti.
	*n*cm	Margine sinistro in centimetri.
	*n*mm	Margine sinistro in millimetri.
	n%	Margine sinistro relativo in percentuale.
margin-right	auto	Margine destro automatico.
	*n*pt	Margine destro in punti.
	*n*cm	Margine destro in centimetri.
	*n*mm	Margine destro in millimetri.
	n%	Margine destro relativo in percentuale.
border-width	thin	Bordo sottile.
	medium	Bordo medio.
	thick	Bordo spesso.
border-color	*colore*	Colore del bordo.
border-style	none	Bordo non visibile.
	dotted	Bordo puntato.
	dashed	Bordo tratteggiato.
	solid	Bordo continuo.
	double	Bordo continuo doppio.
width	auto	Larghezza automatica.
	*n*pt	Larghezza in punti.
	*n*cm	Larghezza in centimetri.
	*n*mm	Larghezza in millimetri.
	n%	Larghezza relativa in percentuale.
height	auto	Altezza automatica.
	*n*pt	Altezza in punti.
	*n*cm	Altezza in centimetri.
	*n*mm	Altezza in millimetri.
	n%	Altezza relativa in percentuale.
float	none	Blocco fisso.
	left	Blocco fluttuante collocato a sinistra con testo che scorre a destra.

Proprietà	Valori	Descrizione
	right	Blocco fluttuante collocato a destra con testo che scorre a sinistra.
clear	none	Scorre normalmente.
	left	Salta un oggetto che si trova a sinistra.
	right	Salta un oggetto che si trova a destra.
	both	Salta qualunque oggetto fluttuante.

55.17.1 File «blocchi-css-01.html»

Si realizzi il file ‘`blocchi-css-01.html`’ con il contenuto seguente, riutilizzando eventualmente esercitazioni già svolte:

```
<!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
<HTML LANG="it">
<HEAD>
    <META HTTP-EQUIV="Content-Type"
          CONTENT="text/html; charset=us-ascii">
    <META NAME="Description"
          CONTENT="Blocchi, esempio 01">
    <META NAME="Keywords"
          CONTENT="HTML, CSS, blocchi, esempio">
    <META NAME="Author"
          CONTENT="Tizio Tizi, tizio@brot.dg, classe 4Z">
    <META NAME="Date" CONTENT="2012.01.01">
    <META NAME="Resource-type" LANG="en"
          CONTENT="Document">
    <META NAME="Revisit-after" LANG="en"
          CONTENT="15 days">
    <META NAME="Robots" CONTENT="ALL">
    <TITLE>Blocchi</TITLE>
    <STYLE TYPE="text/css">
        <!--
            A.sinistra {
                float:          left;
                margin-right:   2cm;
            }
            A.destra {
                float:          right;
                margin-left:    2cm;
            }
            H1 {
                font-weight:    bolder;
                font-size:      2cm;
            }
            P {
                font-family:    sans-serif;
                font-size:      8mm;
                text-align:     justify;
            }
        -->
    </STYLE>
</HEAD>
<BODY>

<H1>Le cose che mi piacciono</H1>

<P><A CLASS="sinistra" HREF="http://www.gnu.org"><IMG
SRC="http://www.gnu.org/graphics/philosophical-gnu-sm.jpg"
ALT="GNU filosofico"></A>Ci sono due cose che mi
piacciono: il software libero e gli standard liberi.
Per saperne di pi&ugrave; sul software libero si
pu&ograve; leggere qualcosa di utile a partire dal sito
http://www.gnu.org (basta fare clic sull'immagine del
"GNU filosofico" che appare qui a sinistra).</P>

<P><A CLASS="destra"
HREF="http://validator.w3.org/check/referer"><IMG
SRC="http://validator.w3.org/images/v15445" ALT="Valid
ISO-HTML!"></A>Per quanto riguarda gli standard, si
veda in particolare il W3C (http://www.w3.org), ma per
verificare la correttezza sintattica di questa pagina
basta fare un bel clic sulla superficie dell'icona che
appare a destra.</P>

</BODY>
</HTML>
```

Figura 55.114. Aspetto del file ‘`blocchi-css-01.html`’.

55.17.2 File «blocchi-css-02.html»

Si realizzi il file ‘`blocchi-css-02.html`’, partendo da una copia di ‘`blocchi-css-01.html`’, in modo da arrivare al contenuto seguente:

```
<!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
<HTML LANG="it">
<HEAD>
    <META HTTP-EQUIV="Content-Type"
          CONTENT="text/html; charset=us-ascii">
    <META NAME="Description"
          CONTENT="Blocchi, esempio 02">
    <META NAME="Keywords"
          CONTENT="HTML, CSS, blocchi, esempio">
    <META NAME="Author"
          CONTENT="Tizio Tizi, tizio@brot.dg, classe 4Z">
    <META NAME="Date" CONTENT="2012.01.01">
    <META NAME="Resource-type" LANG="en"
          CONTENT="Document">
    <META NAME="Revisit-after" LANG="en"
          CONTENT="15 days">
    <META NAME="Robots" CONTENT="ALL">
    <TITLE>Blocchi</TITLE>
    <STYLE TYPE="text/css">
        <!--
            A.sinistra {
                float:          left;
                margin-right:   2cm;
            }
            A.destra {
                float:          right;
                margin-left:    2cm;
            }
            H1 {
                font-weight:    bolder;
                font-size:      15mm;
                margin-left:    2cm;
                margin-right:   2cm;
                margin-top:     2cm;
                margin-bottom:  2cm;
                border-style:   solid;
            }
            P {
                font-family:    sans-serif;
                font-size:      8mm;
                text-align:     justify;
            }
        -->
    </STYLE>
</HEAD>
<BODY>

<H1>Le cose che mi piacciono</H1>

<P><A CLASS="sinistra" HREF="http://www.gnu.org"><IMG
SRC="http://www.gnu.org/graphics/philosophical-gnu-sm.jpg"
ALT="GNU filosofico"></A>Ci sono due cose che mi
piacciono: il software libero e gli standard liberi.
Per saperne di pi&ugrave; sul software libero si
pu&ograve; leggere qualcosa di utile a partire dal sito
http://www.gnu.org (basta fare clic sull'immagine del
"GNU filosofico" che appare qui a sinistra).</P>

<P><A CLASS="destra"
HREF="http://validator.w3.org/check/referer"><IMG
SRC="http://validator.w3.org/images/v15445" ALT="Valid
ISO-HTML!"></A>Per quanto riguarda gli standard, si
veda in particolare il W3C (http://www.w3.org), ma per
verificare la correttezza sintattica di questa pagina
basta fare un bel clic sulla superficie dell'icona che
appare a destra.</P>

</BODY>
</HTML>
```


Figura 55.116. Aspetto del file ‘`blocchi-css-02.html`’.

55.17.3 File «blocchi-css-03.html»

Si realizzi il file ‘`blocchi-css-03.html`’, riutilizzando eventualmente porzioni di esercitazioni già svolte. Il file deve avere il contenuto seguente:

```
<!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
<HTML LANG="it">
<HEAD>
    <META HTTP-EQUIV="Content-Type"
          CONTENT="text/html; charset=us-ascii">
    <META NAME="Description"
          CONTENT="Blocchi, esempio 03">
    <META NAME="Keywords"
          CONTENT="HTML, CSS, blocchi, esempio">
    <META NAME="Author"
          CONTENT="Tizio Tizi, tizio@brot.dg, classe 4Z">
    <META NAME="Date" CONTENT="2012.01.01">
    <META NAME="Resource-type" LANG="en"
          CONTENT="Document">
    <META NAME="Revisit-after" LANG="en"
          CONTENT="15 days">
    <META NAME="Robots" CONTENT="ALL">
    <TITLE>Blocchi</TITLE>
    <STYLE TYPE="text/css">
        <!--
            TABLE {
                border-style:     solid;
                border-width:     thin;
            }
            TH {
                border-style:     none;
                background-color: green;
            }
            TD {
                border-style:     dotted;
                border-width:     thin;
            }
            P {
                margin-top:       5mm;
                margin-bottom:    5mm;
                margin-left:      5mm;
                margin-right:     5mm;
                font-size:        6mm;
            }
        -->
```

```
41      </STYLE>
42  </HEAD>
43  <BODY>
44
45  <H1>Operatori</H1>
46
47  <P>Segue una tabella contenente l'elenco degli operatori
48  matematici pi&ugrave; comuni.</P>
49
50  <TABLE SUMMARY="operatori">
51  <CAPTION>Operatori matematici comuni</CAPTION>
52  <THEAD>
53      <TR>
54      <TH ROWSPAN="2"></TH>
55      <TH ROWSPAN="2"><P>Operatore e operandi</P></TH>
56      <TH COLSPAN="2"><P>Annotazioni</P></TH>
57      </TR>
58      <TR>
59      <TH><P>Operazione</P></TH>
60      <TH><P>Descrizione</P></TH>
61      </TR>
62  </THEAD>
63  <TBODY>
64      <TR>
65      <TH ROWSPAN="4"><P>le quattro operazioni</P></TH>
66      <TD><P><CODE><VAR>op1</VAR>+<VAR>op2</VAR></CODE>
67         </P></TD>
68      <TD><P>somma</P></TD>
69      <TD><P>Somma i due operandi.</P></TD>
70      </TR>
71      <TR>
72      <TD><P><CODE><VAR>op1</VAR>-<VAR>op2</VAR></CODE>
73          </P></TD>
74      <TD><P>sottrazione</P></TD>
75      <TD><P>Sottrae il valore del secondo operando da
76             quello del primo.</P></TD>
77      </TR>
78      <TR>
79      <TD><P><CODE><VAR>op1</VAR>*<VAR>op2</VAR></CODE>
80          </P></TD>
81      <TD><P>moltiplicazione</P></TD>
82      <TD><P>Moltiplica i due operandi.</P></TD>
83      </TR>
84      <TR>
85      <TD><P><CODE><VAR>op1</VAR>/<VAR>op2</VAR></CODE>
86          </P></TD>
87      <TD><P>divisione</P></TD>
88      <TD><P>Divide il primo operando per il valore del
89             secondo.</P></TD>
90      </TR>
91  </TBODY>
92  </TABLE>
93
94  </BODY>
95  </HTML>
```

Figura 55.118. Aspetto del file 'blocchi-css-03.html'.

Operatori

Segue una tabella contenente l'elenco degli operatori matematici più comuni.

Operatori matematici comuni

	Operatore e operandi	Annotazioni	
		Operazione	Descrizione
le quattro operazioni	*op1*+*op2*	somma	Somma i due operandi.
	op1−*op2*	sottrazione	Sottrae il valore del secondo operando da quello del primo.
	*op1***op2*	moltiplicazione	Moltiplica i due operandi.
	op1/*op2*	divisione	Divide il primo operando per il valore del secondo.

55.17.4 Verifica sull'uso delle proprietà riferite ai blocchi

In base a quanto appreso fino a questo punto, si realizzi il file 'verifica-blocchi-css-01.html' che produca un risultato abbastanza simile a quello seguente, quando viene visto attraverso un navigatore:

ASCII

Una piccola porzione del codice ASCII.

Codifica			Carattere
Ottale	Decimale	Esadecimale	
141_8	97	61_{16}	a
142_8	98	62_{16}	b
143_8	99	63_{16}	c
144_8	100	64_{16}	d
145_8	101	65_{16}	e
146_8	102	66_{16}	f
147_8	103	67_{16}	g
150_8	104	68_{16}	h
151_8	105	69_{16}	i
152_8	106	$6A_{16}$	j

Si osservi che il testo nelle celle ha un margine di 5mm e che i valori numerici sono allineati alla destra.

Si completi l'intestazione, coerentemente, secondo il proprio criterio, seguendo le modalità già stabilite. Si deve consegnare per la valutazione:

1. la stampa del risultato ottenuto attraverso il navigatore;
2. la stampa del sorgente.

Si richiede espressamente che tutti gli elementi, a esclusione di quelli che devono essere vuoti, siano terminati correttamente con il marcatore di chiusura.

55.18 Contesto dinamico

Dal momento che un documento HTML viene letto normalmente attraverso un navigatore, con il quale si interagisce, è possibile fare riferimento a delle pseudo-classi il cui ambito riguarda la dinamica di interazione con l'utente. I casi principali riguardano i riferimenti ipertestuali, che possono essere già stati visitati o meno, e la posizione del puntatore del mouse sopra il documento.

Tabella 55.120. Pseudo-classi utili nell'ambito dell'interazione con l'utente.

Pseudo-classe	Descrizione
`:link`	Include i riferimenti ipertestuali che non sono ancora stati visitati.
`:visited`	Include i riferimenti ipertestuali che sono già stati visitati.
`:hover`	Riguarda un componente che si trova a essere sormontato dal puntatore del mouse, senza che l'utente compia altre azioni con questo.
`:active`	Riguarda un componente che si trova a essere selezionato, per esempio con un clic del mouse.

Pseudo-classe	Descrizione
`:focus`	Riguarda un componente che si trova a essere a fuoco, per esempio quando si scrive in una casella di inserimento.

55.18.1 File «dinamica-css-01.html»

Si realizzi il file '`dinamica-css-01.html`' con il contenuto seguente, partendo eventualmente dalla copia di altri esercizi fatti in precedenza:

```
<!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
<HTML LANG="it">
<HEAD>
    <META HTTP-EQUIV="Content-Type"
          CONTENT="text/html; charset=us-ascii">
    <META NAME="Description"
          CONTENT="Pseudo-classi dinamiche, esempio 01">
    <META NAME="Keywords"
          CONTENT="HTML, CSS, pseudo-classi, esempio">
    <META NAME="Author"
          CONTENT="Tizio Tizi, tizio@brot.dg, classe 4Z">
    <META NAME="Date" CONTENT="2012.01.01">
    <META NAME="Resource-type" LANG="en"
          CONTENT="Document">
    <META NAME="Revisit-after" LANG="en"
          CONTENT="15 days">
    <META NAME="Robots" CONTENT="ALL">
    <TITLE>Dinamica</TITLE>
    <STYLE TYPE="text/css">
        <!--
            BODY {
                font-size:        5mm;
            }
            H1 {
                font-size:        1cm;
            }
            SPAN:hover {
                font-size:        1cm;
            }
            A:hover {
                background-color: yellow;
            }
            A:active {
                background-color: red;
            }
            A:link {
                color:            blue;
            }
            A:visited {
                color:            green;
            }
        -->
    </STYLE>
</HEAD>
<BODY>

<H1>Le cose che mi piacciono</H1>

<P>Ci sono due cose che mi piacciono: il <SPAN>software
libero</SPAN> e gli <SPAN>standard liberi</SPAN>. Per
saperne di pi&ugrave; sul <SPAN>software libero</SPAN>
si pu&ograve; leggere qualcosa di utile a partire dal
sito <A
HREF="http://www.gnu.org">http://www.gnu.org</A>; per
quanto riguarda gli standard <SPAN>W3C</SPAN>, si veda
il sito <A
HREF="http://www.w3.org">http://www.w3.org</A>.</P>

</BODY>
</HTML>
```

Non viene mostrato l'aspetto finale del documento, perché ciò che conta è quello che accade quando si sposta il puntatore del mouse sopra ai termini delimitati dagli elementi '**SPAN**' e sugli elementi '**A**'. Si provi anche a selezionare uno dei riferimenti ipertestuali, in modo da vedere cambiare il colore dello sfondo, nel momento della selezione.

55.19 Rispetto del diritto d'autore

Quando si scrive un documento destinato alla diffusione, occorre fare attenzione a rispettare il diritto d'autore e le regole di cortesia comuni. La prima cosa che fa, normalmente, chi comincia a realizzare delle pagine HTML, è quella di recuperare dalla rete ciò che può servire per abbellire il proprio «sito». Dal punto di vista psicologico, questo comportamento sembra una cosa naturale, come sarebbe naturale raccogliere delle conchiglie da una spiaggia o dei sassolini colorati dal letto di un torrente, ma ciò che si trova attraverso Internet non è un dono della natura e occorre un atteggiamento differente.

Sia le leggi sul diritto d'autore, sia le regole di buon comportamento, impongono di accertare la possibilità di riutilizzare del materiale che si potrebbe ottenere attraverso la rete. Una volta accertata la possibilità, salvo altri obblighi stabiliti dall'autore, tutto ciò che non è lavoro proprio deve contenere un riferimento al suo autore e possibilmente all'origine dalla quale questo è stato ottenuto. In circostanze particolari, la pubblicazione di un simbolo o di un logo non richiede l'indicazione del riferimento bibliografico, ma impone probabilmente altri oneri; per esempio, il mostrare il logo della verifica ISO 15445 significa garantire che il proprio lavoro è conforme a tale standard.

Per quanto riguarda la normativa italiana sul rispetto del diritto d'autore, si osservi che non è previsto il *fair use*.

55.19.1 Verifica conclusiva

In base a quanto appreso da queste lezioni su HTML e sui fogli di stile CSS, si realizzi un proprio «sito», scegliendo liberamente il tema. Se si vogliono utilizzare oggetti (immagini o altro) di altri autori, si accerti la possibilità di agire in tal modo, avendo cura di fare i riferimenti del caso. Per quanto riguarda il testo, non si possono copiare brani di documenti di altri autori, salva la possibilità di usare la citazione.

Non si includano file audio-visuali, né software; nel caso di riproduzione di immagini, anche parziali, e di citazioni, ci si accerti di avere ottenuto il permesso (dagli autori, dagli eredi o da chi ne ha acquisito i diritti), oppure che gli autori siano morti da più di 70 anni. Si rammenti che le norme italiane sul diritto d'autore non contemplano il *fair use*.

Il lavoro deve essere conforme allo standard ISO 15445 e alle regole previste per i fogli di stile CSS, pertanto, ogni pagina HTML deve riportare le icone di verifica ISO 15445 e CSS. Si veda eventualmente il file '`stile-01.html`' (sezione 55.12.1), in cui si utilizzano queste due icone.

Il lavoro deve essere accessibile sia attraverso un navigatore grafico, sia attraverso un navigatore senza grafica.

Il lavoro deve riportare il copyright del proprio autore (il copyright deve apparire in ogni file HTML) e possibilmente le condizioni alle quali è sottoposto il suo utilizzo (si veda eventualmente il lavoro di Creative Commons *http://creativecommons.org/*).

55.20 Riferimenti

- W3C, *Technical Reports and Publications*, *http://www.w3.org/TR/*
- W3C, *Cascading Style Sheets*, *http://www.w3.org/TR/CSS1/* , *http://www.w3.org/TR/CSS2/*
- Eric A. Meyer, *CSS/edge*, *http://meyerweb.com/eric/css/edge/*
- *Creative Commons*, *http://creativecommons.org/*

introduzione a RDF: collegamento dei dati nel web

RDF sta per *Resource description framework* e definisce un «linguaggio» per rappresentare informazioni relative a risorse disponibili nel web, allo scopo di realizzare ciò che è noto come «web semantico». Questo lavoro ha origine nel 1999 e i documenti che da allora si sono succeduti utilizzano una terminologia che cambia nel tempo, mano a mano che i concetti astratti originali trovano attuazioni concrete; pertanto, piuttosto che soffermarsi sulla terminologia, è meglio cercare di capire il senso dei vari componenti.

Anche se in relazione a RDF si parla di «linguaggio», in pratica si tratta di un modo di schematizzare le informazioni; tali schemi vengono poi resi in forma testuale in modi diversi, a seconda della necessità o dell'opportunità.

56.1 Un esempio prima di cominciare

Prima è nato il «web», poi sono nati i motori di ricerca, poi si è posto il problema di dare maggiori informazioni ai motori di ricerca per migliorare la classificazione dei file accessibili attraverso la rete e del loro contenuto. A tale proposito, nelle intestazioni delle pagine HTML sono stati usati gli elementi '**META**', per esempio così:

```
<head>
  <meta http-equiv="Content-Type"
        content="text/html; charset=UTF-8">
  <meta name="Author" content="Clara Drusilla">
  <meta name="Date" content="2012-12-31">
  <meta name="Description"
        content="Passatempi di una professoressa">
  <meta name="Keywords"
        content="hobby, tempo libero, mare, spiaggia, sole">
  <title>Il mio hobby preferito</title>
</head>
```

Attualmente è preferibile inserire del codice RDF (per cui si parla di HTML+RDF, o di RDFa), trasformando quanto già mostrato nell'esempio seguente:

```
<head>
  <meta http-equiv="Content-Type"
        property="http://purl.org/dc/terms/format"
        content="text/html; charset=UTF-8">
  <meta property="http://purl.org/dc/terms/title"
        content="Il mio hobby preferito">
  <meta property="http://purl.org/dc/terms/creator"
        content="clara.html">
  <meta property="http://purl.org/dc/terms/created"
        content="2012-12-31">
  <meta property="http://purl.org/dc/terms/description"
        content="Passatempi di una professoressa">
  <meta property="http://purl.org/dc/terms/subject"
        content="hobby, tempo libero, mare, spiaggia, sole">
  <title>Il mio hobby preferito</title>
</head>
```

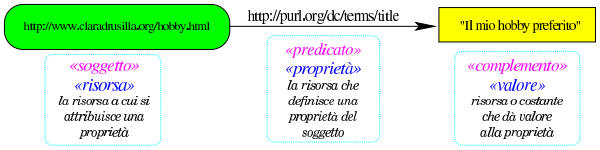
Come si vede, al posto degli attributi '**name**' ci sono attributi

‘**property**’, a cui si assegna un URI che definisce la proprietà dell’attributo ‘**content**’. In pratica, il documento rappresentato dalla pagina HTML che contiene questa intestazione ha il titolo «Il mio hobby preferito», in quanto la proprietà che lo definisce è espressa dall’URI *http://purl.org/dc/terms/title*. Si osservi anche che in questo caso, al posto di indicare il nome dell’autrice (Clara Drusilla), si fa riferimento a un’altra pagina (*clara.html*) che presumibilmente contiene delle informazioni personali più articolate.

56.2 Una base di dati fatta di «triple»

Nel modello delle basi di dati relazionali, i dati sono organizzati in relazioni che si suddividono in tuple; ovvero, si tratta di tabelle suddivise in righe. RDF permette di arrivare alla descrizione di una base di dati che ha la forma di un grafo orientato, attraverso delle ***triple***, ovvero terne di valori. Per la precisione, RDF descrive solo delle triple, con cui si costruisce un grafo più o meno articolato.

Figura 56.3. Una tripla che individua il titolo della pagina *http://www.claradrusilla.org/hobby.html*.



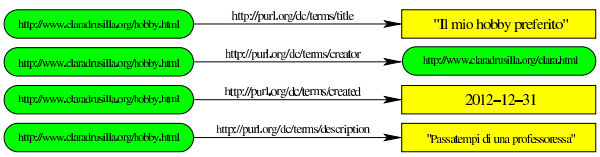
I tre componenti della tripla hanno dei nomi convenzionali che però possono creare confusione, perché non esprimono bene il significato che hanno in pratica: ***soggetto***, ***predicato*** e ***oggetto***. In alternativa, come si vede nell’esempio della figura, possono essere sostituiti ***risorsa***, ***proprietà*** e ***valore***.

1. Il ***soggetto***, ovvero il primo elemento di una tripla, può essere solo un URI (*uniform resource identifier*) e lo scopo della tripla è quello di conoscere il valore di una proprietà di questo soggetto. In casi eccezionali, come viene descritto più avanti nel capitolo, il soggetto può essere rappresentato da un «nodo vuoto», privo di URI.
2. Il ***predicato***, ovvero il secondo elemento della tripla è un URI che definisce qualcosa che riguarda il soggetto e che ha un valore. Questo URI fa parte di solito di un vocabolario, ovvero una raccolta di proprietà. Il verso della freccia che rappresenta la proprietà in una tripla è convenzionale: va sempre indirizzato verso l’oggetto, ovvero il terzo elemento.
3. L’***oggetto***, ovvero il terzo elemento della tripla è il valore della proprietà. Questo valore può essere costituito da un indirizzo URI, oppure da un dato costante, come una stringa, una data o un numero.

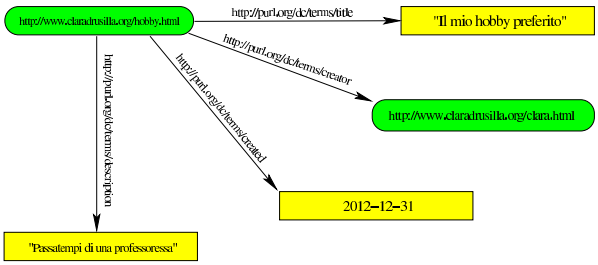
Va osservato che gli indirizzi URI, a cui ci si riferisce in questi casi, non sono necessariamente anche degli indirizzi URL. Pertanto, si tratta di indirizzi univoci, ma a questi indirizzi non fa capo necessariamente una risorsa che possa essere consultabile immediatamente. Va poi osservato anche che, nella realizzazione pratica di RDF, si vedono solo indirizzi che fanno capo al protocollo HTTP.

Un soggetto (il primo elemento di una tripla) può possedere diverse proprietà, ognuna delle quali si esprime con una tripla indipendente, come si vede nella figura successiva.

Figura 56.4. Diverse triple relative a uno stesso soggetto.

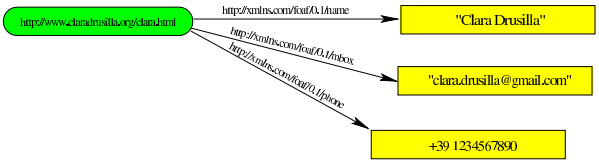


Da questo fatto si può incominciare a intravedere il grafo orientato che si può sviluppare attraverso le triple di RDF:



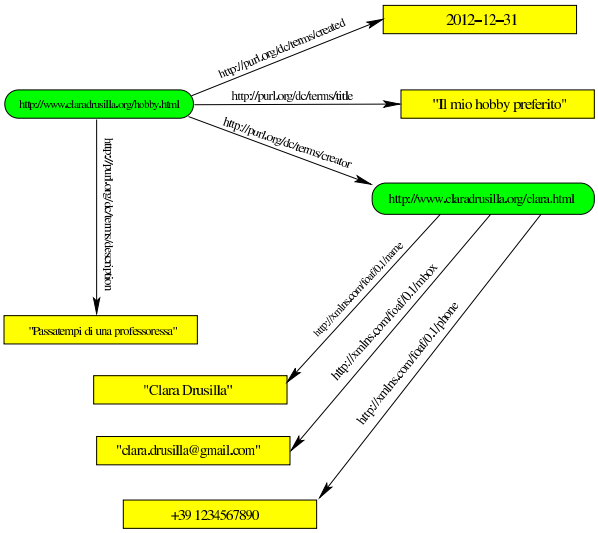
Quando un oggetto (il terzo elemento di una tripla) è costituito da un indirizzo URI, a sua volta questo può essere descritto ulteriormente, diventare soggetto di altre triple.

Figura 56.6. Grafo degli attributi di *http://www.claradrusilla.org/clara.html*.



In presenza di soggetti e oggetti corrispondenti alla stessa risorsa, i grafi possono innestarsi:

Figura 56.7. Innesto dei due grafi.



Rispetto agli esempi mostrati, va aggiunto il fatto che **la proprietà di un soggetto può apparire più volte, con valori diversi**. Per esempio, per indicare più autori di uno stesso soggetto, si può usare la proprietà *http://purl.org/dc/terms/creator* più volte.

56.3 Vocabolari

Ciò che mette in relazione il soggetto con l’oggetto di una tripla (il primo e l’ultimo elemento), ovvero la proprietà, fa parte di un ***vocabolario***. Negli esempi apparsi fino a questo punto, sono stati usati due vocabolari: *http://purl.org/dc/terms/* e *http://xmlns.com/foaf/0.1/*.

Di norma (ma non sempre), se si seguono gli indirizzi URI corrispondenti ai vocabolari rispettivi, attraverso un navigatore ipertestuale, si viene ridiretti alla guida sul loro utilizzo. In tal modo si può conoscere che significato dare a ogni «termine» da usare in qualità di proprietà di una tripla di RDF. Nel caso particolare degli esempi, si può osservare che il primo vocabolario è pensato per attribuire proprietà a ciò che rappresenta un prodotto editoriale (autore, titolo, data,...), mentre il secondo si presta per attribuire proprietà a ciò che rappresenta una persona (nome, indirizzo di posta elettronica, telefono, relazioni con altre persone,...).

Tabella 56.8. Alcuni vocabolari di uso comune.

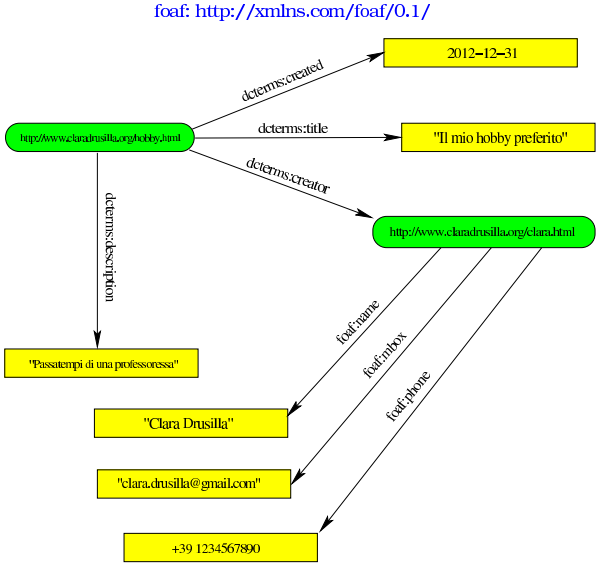
Sigla (*common prefix*)	URI	Descrizione
rdf	*http://www.w3.org/1999/01/22-rdf-syntax-ns#*	Vocabolario RDF.
foaf	*http://xmlns.com/foaf/0.1/*	*friend of a friend* Proprietà relative a risorse che rappresentano persone. L'acronimo si riferisce al fatto che questo vocabolario sia utile per attribuire proprietà a persone menzionate nei blog o nei *social network* (nel senso informatico del termine).
dcterms dct	*http://purl.org/dc/terms/*	*dublin core terms* Proprietà relative a risorse che rappresentano un prodotto editoriale.
dc	*http://purl.org/dc/elements/1.1/*	*dublin core elements* Proprietà relative a risorse che rappresentano un prodotto editoriale.
geo	*http://www.w3.org/2003/01/geo/wgs84_pos#*	Proprietà relative a risorse individuabili geograficamente attraverso delle coordinate.
schema	*http://schema.org/*	«HTML Microdata» trasformato in RDF (*http://www.w3.org/TR/microdata-rdf*).

I vocabolari possono essere costruiti utilizzando il linguaggio «RDF Schema» (da non confondere con *http://schema.org/*). Questo aspetto della definizione dei vocabolari viene omesso nel capitolo ed eventualmente va approfondito nella documentazione ufficiale del consorzio W3C: *www.w3.org/TR/rdf-schema/* .

56.4 Abbreviazioni

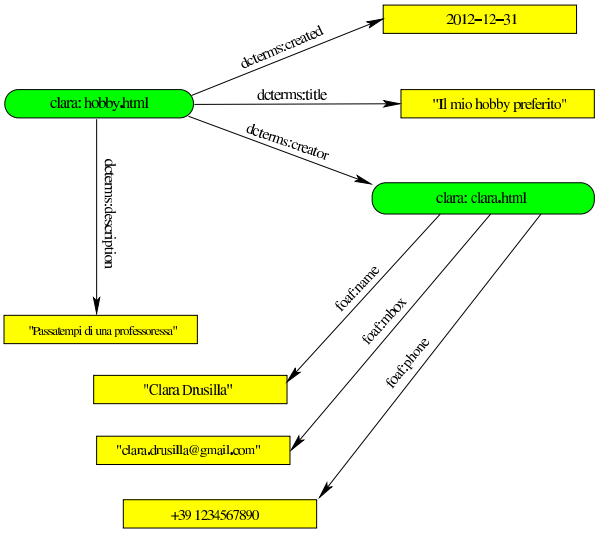
Le risorse che si utilizzano nella costruzione delle triple, possono essere abbreviate. Per fare questo si possono dichiarare delle abbreviazioni come appare nella tabella precedente; per esempio, al posto di scrivere *http://xmlns.com/foaf/0.1/phone* si scriverebbe *foaf:phone*, intendendo la stessa cosa.

Figura 56.9. Rappresentazione di un grafo di triple, abbreviando gli indirizzi delle proprietà.



Anche le risorse che si vanno a descrivere possono essere indicate attraverso delle abbreviazioni, scelte opportunamente in modo da non creare confusione, come si vede nella figura successiva.

Figura 56.10. Abbreviazione anche delle risorse che vengono descritte dalle triple.



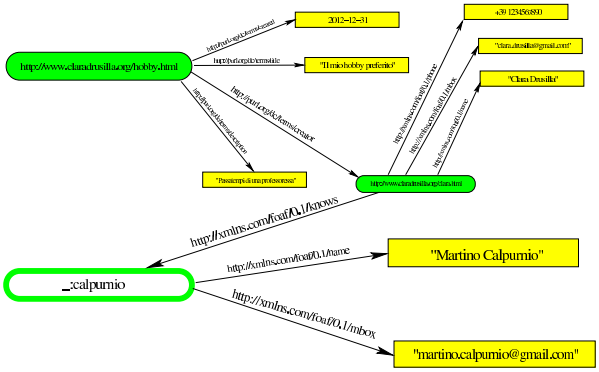
È importante osservare che i percorsi associati ai prefissi, devono essere indicati con attenzione. Per esempio, se il prefisso *foaf* fosse stato indicato come *http://xmlns.com/foaf/0.1*, senza la barra obliqua finale, l'indicazione della risorsa *foaf:name* produrrebbe un risultato errato: *http://xmlns.com/foaf/0.1name* al posto di *http://xmlns.com/foaf/0.1/name*.

Va anche osservato che per poter usare i prefissi, occorre dichiararli espressamente, nella modalità prevista dal contesto in cui si applica il linguaggio RDF. Comunque, in relazione ai vocabolari di proprietà, è sempre bene fare riferimento ai prefissi usati comunemente.

56.5 Nodi vuoti

Ci può essere la necessità o l'opportunità di utilizzare nodi «vuoti», nel senso che non sono identificati da un URI, ma sui quali si vogliono indicare delle proprietà. Per esempio, Clara Drusilla potrebbe voler annotare i dati di alcuni conoscenti nella sua pagina *hobby.html*, come si vede nella figura successiva.

Figura 56.11. Aggiunta di un conoscente con un nodo privo di URI.



Anche se il nodo che si aggiunge non è rappresentato da un URI, occorre dargli un nome provvisorio, in modo da poterlo trattare. Con-

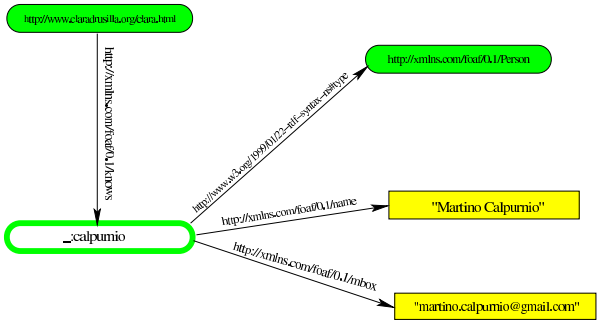
venzionalmente gli si attribuisce il prefisso di abbreviazione '_' (trattino basso), a cui poi si aggiunge un nome qualunque, ma scelto opportunamente, come si vede nella figura. In pratica, in questo modo si definisce un URI inesistente, relativo a un prefisso non dichiarato.

56.6 Tipo di nodo o classe

Quando si dichiarano nodi vuoti, come descritto nella sezione precedente, può essere opportuno specificare che tipo di informazione rappresentano. Il «tipo» di un soggetto si definisce attraverso l'appartenenza a una «classe» secondo RDF. Le classi fanno parte della definizione dei vocabolari di proprietà e, convenzionalmente, i nomi delle classi iniziano con una lettera maiuscola, mentre i nomi delle proprietà hanno una iniziale minuscola.

Per dichiarare espressamente il tipo (la classe) di un soggetto, si usa la proprietà *http://www.w3.org/1999/01/22-rdf-syntax-ns#type*. Nel caso dell'esempio apparso nell'ultima figura, si vede la dichiarazione di un nodo vuoto, relativo a una persona conosciuta da Clara Drusilla: dal momento che questo nodo appare come valore della proprietà *http://xmlns.com/foaf/0.1/knows* si determina implicitamente che deve trattarsi di una persona, secondo la classe *http://xmlns.com/foaf/0.1/Person*, ma lo si può rendere esplicito come si vede nella figura successiva.

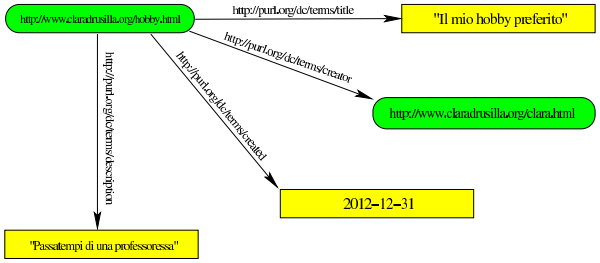
Figura 56.12. Specificazione del tipo di nodo.



56.7 RDFa

La sigla RDFa sta per «RDF in attributes», ovvero una forma di traduzione del linguaggio RDF attraverso attributi SGML/XML. Si tratta in pratica della modalità più semplice di incorporare questo tipo di informazioni in pagine HTML.

Gli attributi del linguaggio RDFa possono essere collocati in qualunque elemento HTML; naturalmente la scelta va fatta in base ai contenuti che si intendono descrivere. All'inizio del capitolo appare un esempio di codice RDFa, con il quale si descrive la pagina *http://www.claradrusilla.org/hobby.html* di Clara Drusilla:




```
<html>
<head>
  <meta http-equiv="Content-Type"
        property="http://purl.org/dc/terms/format"
        content="text/html; charset=UTF-8">
  <meta property="http://purl.org/dc/terms/title"
        content="Il mio hobby preferito">
  <meta property="http://purl.org/dc/terms/creator"
        content="clara.html">
  <meta property="http://purl.org/dc/terms/created"
        content="2012-12-31">
  <meta property="http://purl.org/dc/terms/description"
        content="Passatempi di una professoressa">
  <meta property="http://purl.org/dc/terms/subject"
        content="hobby, tempo libero, mare, spiaggia, sole">
  <title>Il mio hobby preferito</title>
</head>
```


Dal momento che le informazioni riguardano la pagina nel suo complesso, il codice RDFa viene inserito opportunamente nell'intestazione HTML. Si può vedere l'uso dell'attributo '**`property`**', con il quale si indica la risorsa che identifica la proprietà, e l'uso dell'attributo '**`content`**' con il quale si specifica il valore della proprietà. Si osserva che il «soggetto» non viene dichiarato espressamente, pertanto si riferisce alla pagina stessa, ovvero *http://www.claradrusilla.org/hobby.html*.

L'uso dell'attributo '**`content`**' si è reso necessario perché i valori delle proprietà non potevano essere contenuti diversamente: gli elementi '**`META`**' sono elementi vuoti. Supponendo che la pagina degli hobby di Clara Drusilla mostri parte di queste informazioni al lettore, sarebbe stato possibile organizzare gli attributi RDFa in maniera differente:

```
<html>
<head>
  <meta http-equiv="Content-Type"
        property="http://purl.org/dc/terms/format"
        content="text/html; charset=UTF-8">
  <meta property="http://purl.org/dc/terms/creator"
        content="clara.html">
  <meta property="http://purl.org/dc/terms/subject"
        content="hobby, tempo libero, mare, spiaggia, sole">
  <title property="http://purl.org/dc/terms/title">Il mio
    hobby preferito</title>
</head>
<body>
<h1 property="http://purl.org/dc/terms/description">Passatempi
di una professoressa</h1>
<p property="http://purl.org/dc/terms/created">2012-12-31</p>
<p><a href="clara.html">Clara Drusilla</p>
```

In tal modo, il titolo principale, la descrizione della pagina e la data di creazione sono stati presi dai contenuti degli elementi, senza bisogno di inserirli in attributi '**`content`**'.

Per dichiarare espressamente il «soggetto», ovvero la risorsa a cui si sta facendo riferimento, si può usare l'attributo '**`resource`**'. In questo caso viene collocata nell'elemento '**`HTML`**', dato che si tratta effettivamente della pagina:

```
<html resource="hobby.html">
```

Supponendo che Clara Drusilla consenta di copiare la sua pagina in altri siti, ma preferisca che la risorsa a cui si fa riferimento sia sempre quella originale, l'indicazione potrebbe avvenire, opportunamente, in modo completo:

```
<html resource="http://www.claradrusilla.org/hobby.html">
```

Logicamente, l'attributo '**`resource`**' diventa indispensabile quando ci si riferisce a qualcosa che è descritto in un file differente da quello in cui appare; per esempio in una bibliografia, dove si indicano dati editoriali riferiti a documenti esterni. Il campo di azione dell'attributo '**`resource`**' riguarda l'elemento in cui appare; pertanto appare frequentemente in elementi '**`DIV`**' e '**`SPAN`**', all'interno dei quali si descrive la risorsa attraverso proprietà e valori.

Per evitare di dover indicare le proprietà attraverso percorsi completi, è possibile specificare il vocabolario RDF attraverso l'attributo '**`vocab`**':

```
<html vocab="http://purl.org/dc/terms/">
<head>
  <meta http-equiv="Content-Type" property="format"
        content="text/html; charset=UTF-8">
  <meta property="creator" content="clara.html">
  <meta property="subject" content="hobby, tempo libero,
    mare, spiaggia, sole">
  <title property="title">Il mio hobby preferito</title>
```

```
</head>
<body>
<h1 property="description">Passatempi di una
professoressa</h1>
<p property="created">2012-12-31</p>
<p><a href="clara.html">Clara Drusilla</p>
```

In questo caso, tutta la pagina utilizza un solo vocabolario, quindi l'attributo '**vocab**' è stato collocato nell'elemento '**HTML**' complessivo. A ogni modo, le eccezioni al vocabolario predefinito potrebbero essere trattate utilizzando dei percorsi completi, come nell'esempio seguente che si riferisce alla pagina '`clara.html`', nella quale Clara Drusilla mette i propri dati:

```
<html vocab="http://purl.org/dc/terms/">
<head>
  <meta http-equiv="Content-Type" property="format"
        content="text/html; charset=UTF-8">
  <meta property="creator" content="clara.html">
  <meta property="created" content="2012-12-31">
  <meta property="subject" content="dati personali">
  <title property="title">Clara Drusilla</title>
</head>
<body>
<h1 property="description">Scheda personale</h1>
<p>Chi sono: <span
property="http://xmlns.com/foaf/0.1/name">Clara
Drusilla</span></p>
<p>email: <span property="http://xmlns.com/foaf/0.1/mbox">
clara.drusilla@gmail.com</span></p>
<p>tel: <span
property="http://xmlns.com/foaf/0.1/phone">
+39 1234567890</span></p>
```

Quando si usa l'attributo '**vocab**' si crea però ambiguità in merito alla risorsa a cui si sta facendo riferimento; pertanto, anche se si tratta della stessa pagina, è bene specificare anche l'attributo '**resource**', eventualmente vuoto se si vuole proprio la stessa pagina, senza dover sapere come si chiama o dove si colloca:

```
<html resource="" vocab="http://purl.org/dc/terms/">
```

Con l'accortezza di specificare ogni volta anche la risorsa, è possibile dichiarare un vocabolario diverso nell'area che interessa. In questo caso si utilizza un elemento '**DIV**' per isolare la zona:

```
<html resource="" vocab="http://purl.org/dc/terms/">
<head>
  <meta http-equiv="Content-Type" property="format"
        content="text/html; charset=UTF-8">
  <meta property="creator" content="clara.html">
  <meta property="created" content="2012-12-31">
  <meta property="subject" content="dati personali">
  <title property="title">Clara Drusilla</title>
</head>
<body>
<h1 property="description">Scheda personale</h1>
<div resource="" vocab="http://xmlns.com/foaf/0.1/">
<p>Chi sono: <span property="name">Clara Drusilla</span></p>
<p>email: <span
property="mbox">clara.drusilla@gmail.com</span></p>
<p>tel: <span property="phone">+39 1234567890</span></p>
</div>
```

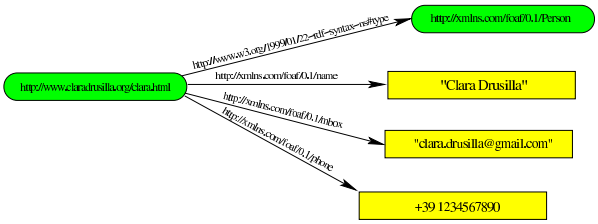
L'uso dell'attributo '**vocab**' consente di non doversi avvalere di abbreviazioni; tuttavia si può fare anche questo, con l'attributo '**prefix**':

```
<html resource="" prefix="dcterms: http://purl.org/dc/terms/
      foaf: http://xmlns.com/foaf/0.1/">
<head>
  <meta http-equiv="Content-Type" property="dcterms:format"
        content="text/html; charset=UTF-8">
  <meta property="dcterms:creator" content="clara.html">
  <meta property="dcterms:created" content="2012-12-31">
  <meta property="dcterms:subject"
        content="dati personali">
  <title property="dcterms:title">Clara Drusilla</title>
</head>
<body>
<h1 property="dcterms:description">Scheda personale</h1>
<p>Chi sono: <span
property="foaf:name">Clara Drusilla</span></p>
<p>email: <span
property="foaf:mbox">clara.drusilla@gmail.com</span></p>
<p>tel: <span
property="foaf:phone">+39 1234567890</span></p>
```


Quando si presenta la necessità o l'opportunità di specificare la classe (il tipo) di appartenenza della risorsa a cui si fa riferimento, si può utilizzare l'attributo '**typeof**', senza bisogno di indicare la proprietà *http://www.w3.org/1999/01/22-rdf-syntax-ns#type*. Anche in occasione dell'uso dell'attributo '**typeof**' è bene precisare la risorsa a cui ci si riferisce, altrimenti si rischia di far intendere che si tratti di un nodo vuoto:

```
<div resource="" vocab="http://xmlns.com/foaf/0.1/"
type="Person">
<p>Chi sono: <span property="name">Clara Drusilla</span></p>
<p>email: <span
property="mbox">clara.drusilla@gmail.com</span></p>
<p>tel: <span property="phone">+39 1234567890</span></p>
</div>
```

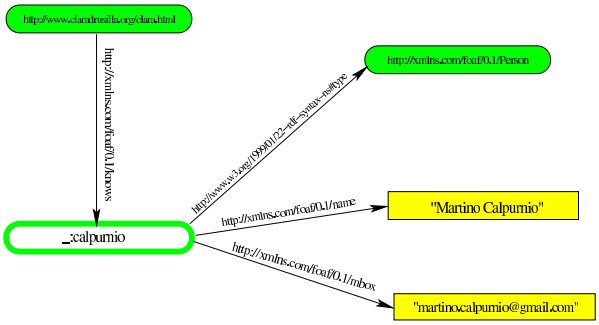
Nell'esempio si vede che nell'area controllata dall'elemento '**DIV**', le proprietà si riferiscono a una risorsa di tipo *Person*. Per la precisione, la classe che definisce il tipo è *http://xmlns.com/foaf/0.1/Person*, essendo stato dichiarato il vocabolario poco prima:



Nell'esempio successivo si vede la creazione di un nodo vuoto in relazione all'indicazione di un conoscente di Clara Drusilla:

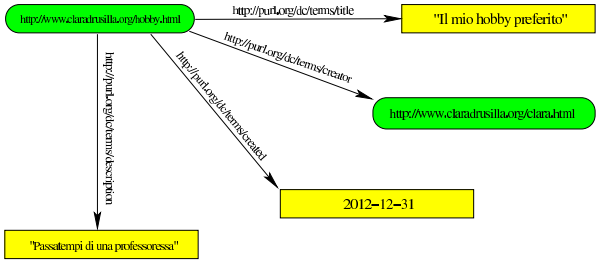
```
<p vocab="http://xmlns.com/foaf/0.1/" property="knows"
type="Person">Il mio migliore amico è <span
property="name">Martino Calpurnio</span> che può essere
contattato all'indirizzo <span
property="mbox">martino.calpurnio@gmail.com</span>.</p>
```

In questo caso, supponendo che prima del codice mostrato si stesse facendo riferimento alla risorsa *http://www.claradrusilla.org/clara.html*, si intende che questa risorsa «conosce» un nodo vuoto, di tipo *http://xmlns.com/foaf/0.1/Person*, il quale ha il nome e l'indirizzo di posta elettronica delimitati con elementi '**SPAN**' appositi:



56.8 N-Triples

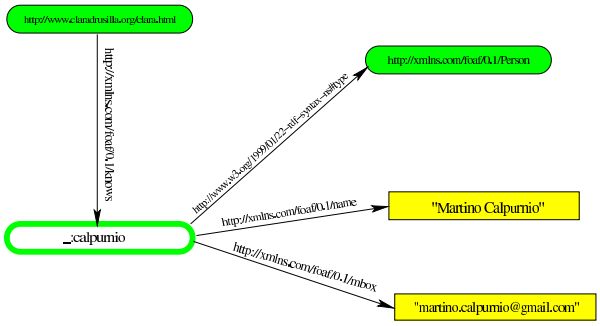
N-Triples è una modalità di rappresentazione molto semplice delle triple RDF; si tratta di righe contenenti i tre elementi consueti: soggetto, predicato e complemento, con la particolarità che si usano le parentesi uncinate, < e >, per delimitare soggetto, predicato ed eventualmente anche l'oggetto quanto questo non è espresso in forma costante. L'esempio seguente mostra la trasformazione di un grafo RDF in N-Triples:




```
# questo è un commento
<http://www.claradrusilla.org/hobby.html> <http://purl.org/dc/terms/title>
    "Il mio hobby preferito" .
<http://www.claradrusilla.org/hobby.html> <http://purl.org/dc/terms/creator>
    <http://www.claradrusilla.org/clara.html> .
<http://www.claradrusilla.org/hobby.html> <http://purl.org/dc/terms/created>
    "2012-12-31" .
<http://www.claradrusilla.org/hobby.html> <http://purl.org/dc/terms/description>
    "Passatempi di una professoressa" .
```

Le righe che compongono le triple, possono essere spezzate, proprio come si vede nell'esempio: in ogni caso, il punto fermo finale determina la conclusione della riga.

I nodi vuoti vengono dichiarati espressamente, attraverso una notazione del tipo '**_:*nome***'. L'esempio seguente traduce il grafo visibile dalla forma grafica alla forma N-Triples:

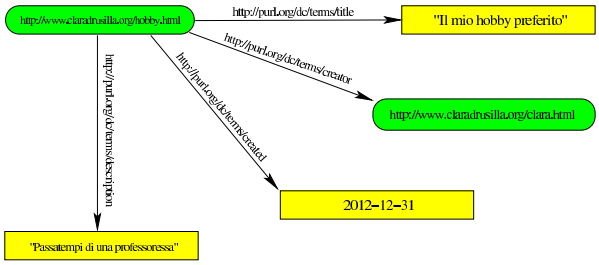



```
# Viene creato un nodo vuoto:
<http://www.claradrusilla.org/clara.html>
    <http://xmlns.com/foaf/0.1/knows> _:calpurnio .
# Vengono definite le proprietà del nodo vuoto:
_:calpurnio <http://www.w3.org/1999/01/22-rdf-syntax-ns#type>
    <http://xmlns.com/foaf/0.1/Person> .
_:calpurnio <http://xmlns.com/foaf/0.1/name>
    "Martino Calpurnio" .
_:calpurnio <http://xmlns.com/foaf/0.1/mbox>
    "martino.calpurnio@gmail.com" .
```

Si osservi che i valori costanti appaiono sempre delimitati come stringhe: per poter specificare il linguaggio o indicare che si tratta invece di valori diversi (per esempio numerici), occorre aggiungere un'estensione. Si veda eventualmente la documentazione elencata alla fine del capitolo.

56.9 Turtle

Turtle sta per *terse RDF triple language* e corrisponde a una modalità di rappresentare il linguaggio RDF in forma testuale, in modo più compatto rispetto a N-Triples. Di fatto, Turtle è un'estensione di N-Triples. Viene ripreso il primo esempio mostrato a proposito di N-Triples, il quale è valido anche per Turtle:




```
# questo è un commento
<http://www.claradrusilla.org/hobby.html> <http://purl.org/dc/terms/title>
    "Il mio hobby preferito" .
<http://www.claradrusilla.org/hobby.html> <http://purl.org/dc/terms/creator>
    <http://www.claradrusilla.org/clara.html> .
<http://www.claradrusilla.org/hobby.html> <http://purl.org/dc/terms/created>
    "2012-12-31" .
<http://www.claradrusilla.org/hobby.html> <http://purl.org/dc/terms/description>
    "Passatempi di una professoressa" .
```

Con Turtle si possono però annotare dei prefissi, così come si può fare con RDFa. L'esempio appena mostrato si potrebbe semplificare nel modo seguente, dopo aver dichiarato i prefissi *clara* e *dcterms*:

```
@prefix clara: http://www.claradrusilla/ .
@prefix dcterms: http://purl.org/dc/terms/ .
clara:hobby.html dcterms:title "Il mio hobby preferito" .
clara:hobby.html dcterms:creator clara:clara.html .
clara:hobby.html dcterms:created "2012-12-31" .
clara:hobby.html dcterms:description
    "Passatempi di una professoressa" .
```

Al posto del prefisso è possibile indicare risorse con percorsi relativi:

```
@base http://www.claradrusilla/ .
@prefix dcterms: http://purl.org/dc/terms/ .
<hobby.html> dcterms:title "Il mio hobby preferito" .
<hobby.html> dcterms:creator clara:clara.html .
<hobby.html> dcterms:created "2012-12-31" .
<hobby.html> dcterms:description
             "Passatempi di una professoressa" .
```

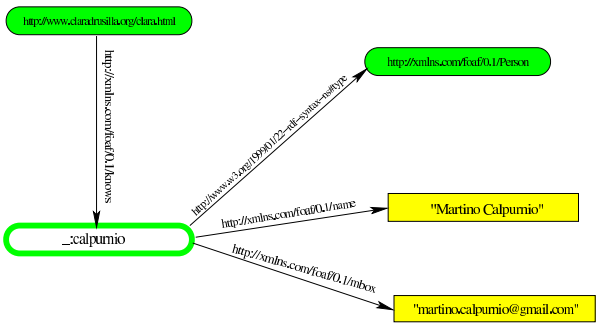
Quando un gruppo di triple condivide lo stesso soggetto, è possibile semplificarle usando il punto e virgola, come si vede nella variante successiva:

```
@base http://www.claradrusilla/ .
@prefix dcterms: http://purl.org/dc/terms/ .
<hobby.html> dcterms:title "Il mio hobby preferito" ;
             dcterms:creator clara:clara.html ;
             dcterms:created "2012-12-31" ;
             dcterms:description
                        "Passatempi di una professoressa" .
```

Se un gruppo di triple condivide lo stesso soggetto e lo stesso predicato, è possibile semplificarle usando la virgola, come si vede nella variante successiva, dove si immagine di avere due descrizioni alternative per la pagina degli hobby della professoressa Clara Drusilla:

```
@base http://www.claradrusilla/ .
@prefix dcterms: http://purl.org/dc/terms/ .
<hobby.html> dcterms:title "Il mio hobby preferito" ;
             dcterms:creator clara:clara.html ;
             dcterms:created "2012-12-31" ;
             dcterms:description
                        "Passatempi di una professoressa" ,
                        "Rilassarsi all'aria aperta" .
```

I nodi vuoti vengono annotati nello stesso modo già visto a proposito di N-Triples, ma con le estensioni di Turtle:




```
@base http://www.claradrusilla/ .
@prefix foaf: http://xmlns.com/foaf/0.1/ .
# Viene creato un nodo vuoto:
<clara.html> foaf:knows _:calpurnio .
# Vengono definite le proprietà del nodo vuoto:
_:calpurnio <http://www.w3.org/1999/01/22-rdf-syntax-ns#type>
            foaf:Person ;
            foaf:name "Martino Calpurnio" ;
            foaf:mbox "martino.calpurnio@gmail.com" .
```

Con Turtle si può rimpiazzare la proprietà *http://www.w3.org/1999/01/22-rdf-syntax-ns#type* con la sigla '**a**':

```
@base http://www.claradrusilla/ .
@prefix foaf: http://xmlns.com/foaf/0.1/ .
# Viene creato un nodo vuoto:
<clara.html> foaf:knows _:calpurnio .
# Vengono definite le proprietà del nodo vuoto:
_:calpurnio a         foaf:Person ;
            foaf:name "Martino Calpurnio" ;
            foaf:mbox "martino.calpurnio@gmail.com" .
```

56.10 Riferimenti

- Network Working Group, *RFC 3987: Internationalized Resource Identifiers (IRIs)*, 2005, *http://www.ietf.org/rfc/rfc3987.txt*
- W3C, *RDF*, 2004, *http://www.w3.org/TR/rdf-syntax/*, *http://www.w3.org/TR/rdf-concepts/*, *http://www.w3.org/TR/rdf-syntax-grammar/*, *http://www.w3.org/TR/rdf-mt/*, *http://www.w3.org/TR/rdf-schema/*, *http://www.w3.org/TR/rdf-testcases/*
- W3C, *RDFa*, 2012, *http://www.w3.org/TR/rdfa-primer/*, *http://www.w3.org/TR/rdfa-core/*, *http://www.w3.org/TR/rdfa-lite/*, *http://www.w3.org/TR/rdfa-in-html/*, *http://www.w3.org/TR/xhtml-rdfa/*
- W3C, *Turtle*, *http://www.w3.org/TR/turtle/*
- W3C, *RDFa validator*, *http://www.w3.org/2012/pyRdfa/Validator.html*
- RDF:about, *RDF validator and converter*, *http://www.rdfabout.com/demo/validator/*
- W3C, *SPARQL*, *http://www.w3.org/TR/rdf-sparql-query/*, *http://www.w3.org/TR/sparql11-query/*

Spartiti musicali

57.1 Concetti elementari sulle notazioni musicali

La notazione musicale odierna, utilizza un pentagramma, ovvero cinque linee e quattro spazi orizzontali, sul quale si dispongono dei simboli che rappresentano principalmente la frequenza di suoni e la loro durata. Secondo le convenzioni, i suoni sono rappresentati da note, a cui sono attribuiti dei nomi, delimitando un intervallo di frequenze chiamato ottava. Un'ottava rappresenta solo un intervallo limitato di queste frequenze; l'ottava successiva è costituita da frequenze doppie rispetto all'ottava precedente; in pratica, una nota di un'ottava ha una frequenza doppia rispetto alla stessa nota nell'ottava precedente.

57.1.1 Denominazione delle note

Secondo le convenzioni attuali, le note sono sette, con nomi diversi a seconda della lingua utilizzata. Tuttavia, queste sette note rappresentano 12 semitoni.

Esistono due denominazioni: «DO, RE, MI, FA, SOL, LA, SI», stabilita attorno all'anno 1000 da Guido d'Arezzo; «C, D, E, F, G, A, B», di origine greca antica. Attualmente, la seconda notazione è quella utilizzata nella lingua inglese e, con una piccola variante, nella lingua tedesca.

Tabella 57.1. Denominazione delle note musicali.

Frequenza di riferimento	italiano, spagnolo, francese	inglese, olandese	tedesco, norvegese, svedese
494 Hz	SI	B	H
440 Hz	LA	A	A
392 Hz	SOL	G	G
349 Hz	FA	F	F
330 Hz	MI	E	E
294 Hz	RE	D	D
262 Hz	DO	C	C

Secondo le convenzioni, la frequenza di riferimento è quella della nota LA, a 440 Hz. Da questa frequenza si ottengono quelle degli altri semitoni dell'ottava, secondo la formula: 440*k^n. La costante k vale precisamente $2^{1/12}$; pertanto, i 12 semitoni corrispondono alla tabella successiva:

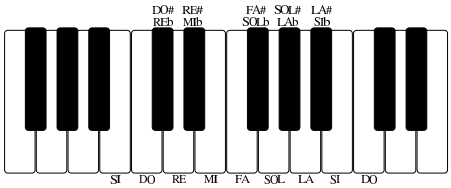
frequenza di riferimento	calcolo	italiano, spagnolo, francese	inglese	olandese	tedesco	norvegese, svedese
494 Hz	$440*k^{2}$	SI, *DOb*	B, *C flat*	B, *Ces*	H, *Ces*	H, *Cess*
466 Hz	$440*k^{1}$	LA#, SIb	A sharp, B flat	Ais, Bes	Ais, B	Aiss, B
440 Hz	$440*k^{0}$	LA	A	A	A	A
415 Hz	$440*k^{-1}$	SOL#, LAb	G sharp, A flat	Gis, Aes	Gis, As	Giss, Aess
392 Hz	$440*k^{-2}$	SOL	G	G	G	G
370 Hz	$440*k^{-3}$	FA#, SOLb	F sharp, G flat	Fis, Ges	Fis, Ges	Fiss, Gess
349 Hz	$440*k^{-4}$	FA, *MI#*	F, *E sharp*	F, *Eis*	F, *Eis*	F, *Eiss*
330 Hz	$440*k^{-5}$	MI, *FAb*	E, *F flat*	E, *Fes*	E, *Fes*	E, *Fess*
311 Hz	$440*k^{-6}$	RE#, MIb	D sharp, E flat	Dis, Es	Dis, Es	Diss, Ess
294 Hz	$440*k^{-7}$	RE	D	D	D	D
277 Hz	$440*k^{-8}$	DO#, REb	C sharp, D flat	Cis, Des	Cis, Des	Ciss, Dess
262 Hz	$440*k^{-9}$	DO, *SI#*	C, *B sharp*	C, *His*	C, *His*	C, *Hiss*

In base a questa formuletta, si può verificare facilmente che la frequenza di una nota in un'ottava diventa doppia nell'ottava successiva. Per esempio, se il DO corrisponde a $440*k^{-9}$, ovvero 262 Hz nell'ottava di riferimento, nell'ottava successiva corrisponde a $440*k^{-9+12} = 440*k^{4}$, pari a 524 Hz.

Tra le note MI e FA c'è solo lo spazio di un semitono. Nello stesso modo, tra il SI e il DO successivo, c'è solo un semitono. In tutti gli altri casi, la distanza tra le note è di un tono (due semitoni).

La scelta di distribuire le note in questo modo, deriva dall'esigenza di distinguere le ottave negli strumenti a tastiera (come il piano). La figura 57.3 evidenzia un'ottava su una porzione di tastiera musicale.

Figura 57.3. Distribuzione di un'ottava su uno strumento a tastiera.

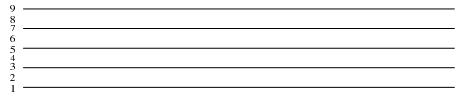


A seconda del punto di vista, quando tra due note c'è lo spazio di uno tono intero, si può indicare il semitono mancante con il nome della nota che lo precede, seguito dal simbolo «diesis», che assomiglia a un cancelletto, oppure con la nota successiva, seguita dal simbolo «bemolle», che assomiglia a una lettera «b» minuscola. Negli strumenti a tastiera non c'è differenza tra diesis della nota precedente e bemolle della nota successiva, mentre negli strumenti ad arco si intende indicare un suono che è un po' più vicino all'una o all'altra nota, senza essere precisamente nel mezzo.

57.1.2 Disposizione simbolica delle note sul pentagramma

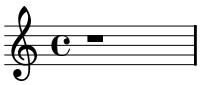
La nota è rappresentata da un simbolo disposto sul pentagramma: la posizione di questo simbolo in relazione alle righe del pentagramma indica la nota stessa; la forma del simbolo rappresenta invece la durata della nota.

Il pentagramma si disegna con cinque linee orizzontali, che delimitano in pratica nove righe su cui collocare le note musicali (cinque linee e quattro spazi):



La corrispondenza di queste righe con le note dipende da un simbolo iniziale, speciale, denominato «chiave». La chiave che consente di rappresentare facilmente le note dell'ottava che si usa comunemente è la chiave di violino, o chiave di SOL (perché centrata sulla seconda linea, dal basso, a rappresentare la nota SOL).

Figura 57.5. Pentagramma con chiave di SOL iniziale.



In condizioni normali, le righe del pentagramma fanno riferimento alla sequenza delle sette note principali, con il SOL centrato sulla riga corrispondente alla seconda linea, come si vede nella figura successiva.

Figura 57.6. Posizioni sul pentagramma, secondo la disposizione normale con chiave di SOL.

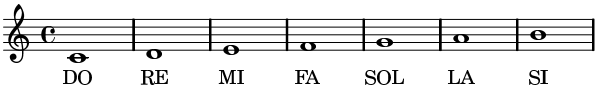
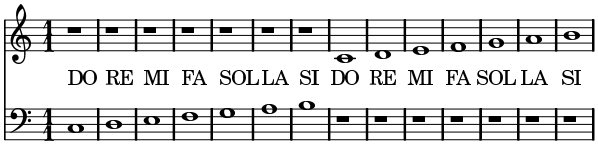


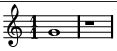
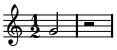
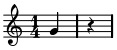
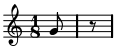
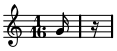
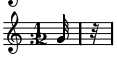
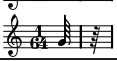
Figura 57.7. Posizioni sul pentagramma, secondo la disposizione normale con chiave di basso e chiave di SOL. Si osservi che le prime sette note sono esattamente nell'ottava inferiore rispetto a quelle che compaiono in riferimento alla chiave di SOL.



57.1.3 Rappresentazione delle note e delle pause sul pentagramma

Il simbolo usato per rappresentare una nota sul pentagramma è un ovale (***testa***), leggermente obliquo, che può essere vuoto, pieno, può avere una ***gamba***, ovvero una stanghetta, che si innalza o si abbassa indifferentemente a seconda della collocazione della nota. La forma del simbolo determina la durata, che si esprime come frazione, oppure attraverso un nome particolare. Oltre alle note, nel pentagramma si indicano pause di silenzio, come si vede riassunto nella tabella successiva.

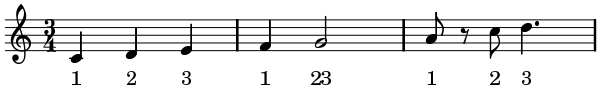
Tabella 57.8. Rappresentazione simbolica della durata delle note e delle pause di silenzio.

Nota e pausa	Durata	Nome
	Intero	Semibreve
	Metà	Minima
	Quarto	Semiminima
	Ottavo	Croma
	Sedicesimo	Semicroma
	Trentaduesimo	Biscroma
	Sessantaquattresimo	Semibiscroma

La durata di una nota può essere allungata della metà del suo valore, aggiungendo un punto subito dopo la testa della stessa. Per esempio, una nota da un quarto (semiminima) seguita da un punto, vale 3/8 (1/4 + 1/8); nello stesso modo, una nota da un quarto, seguita da due punti in orizzontale, vale 7/16 (1/4 + 1/8 + 1/16).

57.1.4 Unità di tempo

Lo spazio orizzontale del pentagramma viene suddiviso in porzioni, denominate ***battuta*** o ***misura***, che rappresentano una durata uniforme. All'interno di queste battute, possono essere inserite note e pause, per un totale conforme al ***tempo***, che si esprime come frazione, all'inizio del pentagramma stesso.



L'esempio mostra un tempo di 3/4, che sta a indicare una misura contenente un totale di tre semiminime (tre quarti).

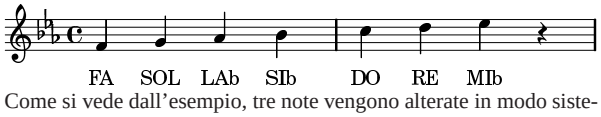
Naturalmente, la durata effettiva dipende dalla lunghezza della battuta.

57.1.5 Alterazioni

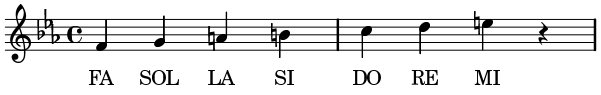
Il valore di una nota dipende dalla collocazione verticale che ha sul pentagramma, tenendo conto della chiave iniziale, che stabilisce il riferimento di partenza. Dal momento che in questo modo manca la possibilità di rappresentare alcuni semitoni, si devono usare delle rappresentazioni speciali per questi. Si usano precisamente i simboli della tabella seguente:

Simbolo	Nome in italiano	Nome in inglese	Significato
♯	diesis	sharp	Innalza di un semitono.
𝄪	doppio diesis	double sharp	Innalza di due semitoni.
𝄪	doppio diesis	double sharp	Innalza di due semitoni.
♭	bemolle	flat	Abbassa di un semitono.
𝄫	doppio bemolle	double flat	Abbassa di due semitoni.
♮	bequadro	natural	Ripristina il tono normale.

Questi simboli si collocano sul pentagramma, per indicare che ciò che segue viene alterato di conseguenza, ma il campo di azione è diverso a seconda del contesto: se si inseriscono all'inizio, subito dopo la chiave, riguardano tutto il pentagramma; se si inseriscono nell'ambito di una battuta, riguardano le note successive, fino alla fine della battuta.



Come si vede dall'esempio, tre note vengono alterate in modo sistematico: MI, che si deve intendere come MI-bemolle; LA, che si deve intendere come LA-bemolle; SI, che si deve intendere SI-bemolle. Volendo invece ripristinare occasionalmente i valori normali, si usa il bequadro:



57.2 Introduzione a LilyPond

LilyPond[1] è una sorta di compilatore, per la composizione musicale. Per la composizione si parte da un sorgente, scritto con un linguaggio che ha delle similitudini con TeX/LaTeX, il quale viene compilato per generare lo spartito in formato DVI. La compilazione, se richiesto, può generare anche un file MIDI.

LilyPond è un lavoro in continuo sviluppo, a causa del quale il formato del sorgente subisce cambiamenti da una versione all'altra.

Qui vengono descritti alcuni concetti essenziali riferiti all'utilizzo di questo programma, con alcuni esempi molto semplici, con l'auspicio che quanto mostrato rimanga valido il più a lungo possibile. Fortunatamente, la documentazione originale di LilyPond è abbastanza buona e offre la descrizione di esempi aggiornati.

57.2.1 Note e pause

Senza entrare troppo nel dettaglio dell'articolazione del sorgente di LilyPond, dal momento che questo è sempre soggetto a variazioni, si può cominciare con una struttura molto semplice, secondo il modello seguente:

```
\include "linguaggio.ly"
\score {
    \notes {
        note
    }
    \paper {
        impaginazione
    }
}
```

Quelle che si vedono sembrano delle macro di TeX, ma in pratica non viene usata qui tale definizione.

Inizialmente, si vede l'istruzione '**\include**', che intuitivamente serve a incorporare uno stile esterno. La notazione musicale cambia da un paese all'altro e quella predefinita è olandese. Pertanto, nella maggior parte delle situazioni, diventa necessario incorporare inizialmente lo stile che consente di utilizzare la notazione più adatta alla propria nazionalità.

Nell'ambito della direttiva '**\notes**', si indicano le note con delle sigle che cambiano in base alla scelta della nazionalità, secondo la tabella 57.13.

Tabella 57.13. Denominazione delle note in base alla nazionalità.

Stile	DO	RE	MI	FA	SOL	LA	SIb	SI	Diesis	Bemolle	Pausa
nederlands.ly	c	d	e	f	g	a	bes	b	xis	xes	r
english.ly	c	d	e	f	g	a	bf bflat	b	xs xsharp	xf xflat	r
deutsch.ly	c	d	e	f	g	a	b	h	xis	xes	r
norsk.ly	c	d	e	f	g	a	b	h	xiss xis	xess xes	r
svenska.ly	c	d	e	f	g	a	b	h	xiss	xess	r
italiano.ly	do	re	mi	fa	sol	la	sib	si	xd	xb	r
catalano.ly	do	re	mi	fa	sol	la	sib	si	xd xs	xb	r
español.ly	do	re	mi	fa	sol	la	sib	si	xs	xb	r

Si noti che la pausa si indica con la lettera «r», perché questa è l'iniziale di *rest*.

Gli esempi vengono mostrati con la notazione per la lingua italiana, pertanto, il modello iniziale per gli esempi può essere quello seguente:

```
\include "italiano.ly"
\score {
    {
        note
    }
    \layout {}
}
```

Per cominciare si può scrivere la sequenza delle note naturali per vedere poi cosa si ottiene:

```
% primo.ly
\include "italiano.ly"
\score {
    {
        do re mi fa sol la si r
    }
    \layout {}
}
```

Supponendo di chiamare il file ‘`primo.ly`’, come si vede dal commento iniziale in stile TeX, si può procedere alla sua compilazione nel modo seguente:

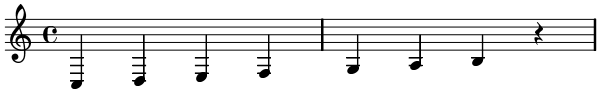
```
$ lilypond --dvi primo.ly [Invio]
```

Durante la compilazione, si possono vedere i messaggi seguenti:

```
GNU LilyPond 2.4.5
Processing ‘primo.ly’
Parsing...
Interpreting music... [3]
Preprocessing graphical objects...
Calculating line breaks...
Layout output to ‘primo.tex’...

Converting to ‘primo.dvi’...
```

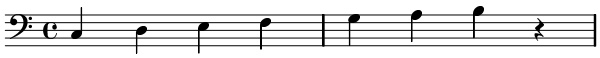
Da quello che si vede, si comprende che viene generato il file ‘`primo.tex`’, compilato successivamente con LaTeX. Si ottiene così un file in formato DVI, che può essere visualizzato, o stampato, con gli strumenti appropriati (sezione 26.3). Si dovrebbe ottenere un risultato simile a quello della figura successiva:



È stata attribuita una chiave di violino in modo predefinito, con un tempo di un intero; inoltre, le note partono da un’ottava al di sotto rispetto a quella «media» della chiave utilizzata e la loro durata predefinita è di un quarto.

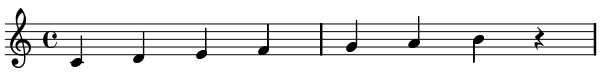
Si può definire esplicitamente la chiave con l’istruzione ‘**`\clef`**’. Le chiavi più comuni sono la chiave di violino, ‘**`\clef treble`**’, e la chiave di basso, ‘**`\clef bass`**’. Viene riproposto l’esempio precedente, usando però una chiave di basso:

```
\include "italiano.ly"
\score {
    {
        \clef bass
        do re mi fa sol la si r
    }
    \layout {}
}
```

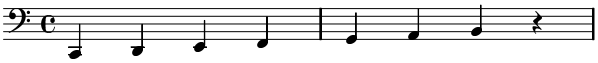


Il tono di una nota può essere alzato facendola seguire da un apice singolo: ogni apice alza la nota di un’ottava; nello stesso modo, si può abbassare il tono di una o più ottave, con l’uso di una virgola. Seguono due esempi:

```
\include "italiano.ly"
\score {
    {
        \clef treble
        do re mi fa sol la si r
    }
    \layout {}
}
```

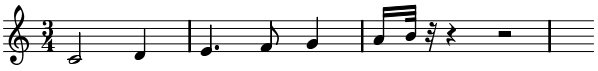



```
\include "italiano.ly"
\score {
    {
        \clef bass
        do, re, mi, fa, sol, la, si, r
    }
    \layout {}
}
```



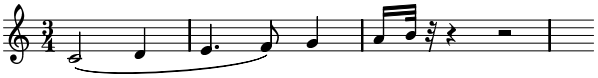
Il tempo si definisce con l’istruzione ‘**`\time`**’, seguita dalla frazione a cui si vuole fare riferimento (per esempio ‘**`3/4`**’). La lunghezza predefinita di una nota è di un quarto. In generale, si definisce il denominatore della frazione di tempo attribuita a una nota o a una pausa con un numero intero che segue la definizione della nota stessa. Inoltre, per prolungare del 50 % la durata della nota, si può aggiungere un punto. L’esempio seguente riassume questi concetti:

```
\include "italiano.ly"
\score {
    {
        \clef treble
        \time 3/4
        do’2 re’4 mi’4. fa’8 sol’4 la’16 si’32 r32 r4 r2
    }
    \layout {}
}
```



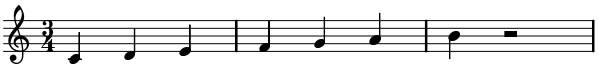
Per definire un legato, si usano le parentesi tonde, ma ciò subito dopo la prima nota e subito prima dell’ultima da unire assieme:

```
\include "italiano.ly"
\score {
    {
        \clef treble
        \time 3/4
        do’2 ( re’4 mi’4. ) fa’8 sol’4 la’16 si’32 r32 r4 r2
    }
    \layout {}
}
```



Per facilitare l’introduzione delle note, è possibile indicare le note all’interno dell’istruzione ‘**`\relative`**’, con cui si dichiara il DO di partenza dell’ottava a cui si vuole fare riferimento:

```
#(ly:set-option ’old-relative)
\include "italiano.ly"
\score {
    {
        \clef treble
        \time 3/4
        \relative do’ {
            do4 re4 mi4 fa4 sol4 la4 si4 r2
        }
    }
    \layout {}
}
```

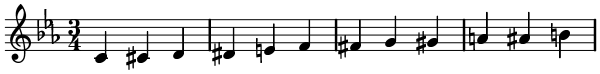


57.2.2 Alterazioni

Come mostra la tabella 57.13, si può alterare il tono di una nota aggiungendo una terminazione, che cambia in base allo stile nazionale selezionato. È poi LilyPond che mostra i simboli correttivi necessari (gli «accidenti»), nel modo più opportuno.

Per definire una scala predefinita differente da quella naturale in DO maggiore, si può usare l’istruzione ‘**`\key`**’, seguita dagli attributi appropriati. L’esempio seguente mostra le note di una «scala cromatica», composta quindi dai 12 semitoni di un’ottava, con la dichiarazione iniziale di una scala in DO minore.

```
#(ly:set-option 'old-relative)
\include "italiano.ly"
\score {
    {
        \clef treble
        \key do \minor
        \time 3/4
        \relative do' {
            do4 dod4 re4 red4 mi4 fa4 fad4 sol4 sold4 la4
            lad4 si4
        }
    }
    \layout {}
}
```



57.2.3 Impaginazione

L'istruzione '**\layout**', che è già stata usata negli esempi, ma sempre con un argomento nullo, controlla l'impaginazione. In modo predefinito è definito il formato della carta e un'intestazione contenente il numero di pagina. L'esempio seguente mostra come controllare il formato della carta e come eliminare la numerazione delle pagine:

```
    \layout {
        papersize = "a4"
        pagenumber = "no"
    }
```

LilyPond consente anche di definire il titolo, l'autore e l'intestazione dei fogli che vanno a contenere lo spartito. Questo si ottiene nell'ambito dell'istruzione '**\header**', che si colloca al di fuori di '**\score**'. Si osservi l'esempio seguente:

```
\include "italiano.ly"
\header {
    title   = "Scala in DO maggiore"
    author  = "Tizio Tizi"
    tagline = "La mia scala"
}
\score {
    …
}
```

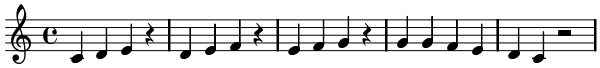
Se non si utilizza la definizione dell'intestazione dei fogli, con l'assegnamento della variabile '**tagline**', LilyPond ne inserisce una predefinita; pertanto, se si vuole togliere questa intestazione, occorre assegnare una stringa nulla:

```
\header {
    tagline = ""
}
```

57.2.4 File MIDI

È possibile ottenere un file in formato MIDI dalla compilazione di un file LilyPond, purché esista l'istruzione '**\midi**', che si colloca solitamente vicino a '**\layout**'. L'istruzione '**\midi**' vuole un argomento, che solitamente è un gruppo vuoto ('**{}**'), a meno che si debbano indicare delle caratteristiche particolari. L'esempio seguente genera anche un file con estensione '`.midi`', contenente l'esecuzione dello spartito. Dal momento che non viene specificato lo strumento, si ottiene il suono del piano.


```
#(ly:set-option 'old-relative)
\include "italiano.ly"
\score {
    {
        \clef treble
        \key do \major
        \time 4/4
        \relative do' {
            do re mi r
            re mi fa r
            mi fa sol r
            sol sol fa mi
            re do r2
        }
    }
    \layout {}
    \midi {}
}
```



Per generare un suono diverso dal piano, si deve intervenire con un'istruzione particolare nell'ambito di '**\score**'. L'estratto seguente è una variante dell'esempio appena mostrato, in cui si dichiara espressamente che lo strumento abbinato è il clavicembalo:

```
\score {
    {
        \set Staff.midiInstrument = harpsichord
        \clef treble
        \key do \major
        \time 4/4
        \relative do' {
            …
        }
    }
    \layout {}
    \midi {}
}
```

I nomi degli strumenti che si possono indicare sono prestabiliti, come riportato nella tabella 57.37.

Tabella 57.37. Elenco dei nomi degli strumenti, secondo la documentazione di LilyPond.

acoustic grand	bright acoustic	electric grand	honky-tonk
electric piano 1	electric piano 2	harpsichord	clav
celesta	glockenspiel	music box	vibraphone
marimba	xylophone	tubular bells	dulcimer
drawbar organ	percussive organ	rock organ	church organ
reed organ	accordion	harmonica	concertina
acoustic guitar (nylon)	acoustic guitar (steel)	electric guitar (jazz)	electric guitar (clean)
electric guitar (muted)	overdriven guitar	distorted guitar	guitar harmonics
acoustic bass	electric bass (finger)	electric bass (pick)	fretless bass
slap bass 1	slap bass 2	synth bass 1	synth bass 2
violin	viola	cello	contrabass
tremolo strings	pizzicato strings	orchestral strings	timpani
string ensemble 1	string ensemble 2	synthstrings 1	synthstrings 2
choir aahs	voice oohs	synth voice	orchestra hit
trumpet	trombone	tuba	muted trumpet
french horn	brass section	synthbrass 1	synthbrass 2
soprano sax	alto sax	tenor sax	baritone sax
oboe	english horn	bassoon	clarinet
piccolo	flute	recorder	pan flute
blown bottle	skakuhachi	whistle	ocarina
lead 1 (square)	lead 2 (sawtooth)	lead 3 (calliope)	lead 4 (chiff)
lead 5 (charang)	lead 6 (voice)	lead 7 (fifths)	lead 8 (bass+lead)
pad 1 (new age)	pad 2 (warm)	pad 3 (polysynth)	pad 4 (choir)
pad 5 (bowed)	pad 6 (metallic)	pad 7 (halo)	pad 8 (sweep)

fx 1 (rain)	fx 2 (soundtrack)	fx 3 (crystal)	fx 4 (atmosphere)
fx 5 (brightness)	fx 6 (goblins)	fx 7 (echoes)	fx 8 (sci-fi)
sitar	banjo	shamisen	koto
kalimba	bagpipe	fiddle	shanai
tinkle bell	agogo	steel drums	woodblock
taiko drum	melodic tom	synth drum	reverse cymbal
guitar fret noise	breath noise	seashore	bird tweet
telephone ring	helicopter	applause	gunshot

Per indicare a quale velocità eseguire lo spartito nella conversione MIDI, all'interno dell'istruzione '**\midi**' si può inserire l'istruzione seguente:

```
\tempo n = m
```

In questo modo si intende una velocità di esecuzione che permetta di avere ***m*** note della durata di 1/***n*** in un minuto. L'esempio seguente serve ad avere 84 note da un quarto al minuto:

```
\midi {
    \tempo 4 = 84
}
```

Esiste anche la possibilità di estrapolare lo spartito da un file MIDI; tuttavia, il risultato che si ottiene risulta privo di indicazioni importanti, o peggio, se il file è stato ottenuto da un'esecuzione vera e propria, i tempi risultano imprecisi. In generale, questa possibilità consente di ottenere una bozza di sorgente LilyPond da adattare successivamente. Supponendo di disporre del file '`prova.midi`', si ottiene il sorgente '`prova-midi.ly`' con il comando seguente:

```
$ lilypond -m prova.ly [Invio]
```

57.2.5 Altri formati

Dal momento che esistono diverse versioni, anche incompatibili, del formato del sorgente di LilyPond, si può usare il comando '**convert-ly**' per aggiornare una vecchia versione di questo sistema di scrittura musicale. Tuttavia, fino a questo punto è stata omessa l'istruzione '**\version**', con cui si specifica, all'inizio del sorgente di LilyPond, a quale versione si sta facendo riferimento:

```
\version 1.5.65
\score {
…
}
```

È sufficiente avviare il programma '**convert-ly**' con il nome del sorgente da convertire, per ottenere l'aggiornamento dello stesso all'ultima versione disponibile (installata) di LilyPond, attraverso lo standard output. Eventualmente, si può usare l'opzione '**-e**', per aggiornare il file di partenza (lasciando una copia di sicurezza della versione precedente con estensione '~'):

```
$ convert-ly -e prova.ly [Invio]
```

In questo esempio, si ottiene quindi un nuovo file '`prova.ly`', lasciando una copia di quello vecchio con il nome '`prova.ly~`'.

Sono disponibili anche dei programmi di conversione ulteriori, per trasformare altri formati in LilyPond: '**etf2ly**' '**abc2ly**' '**pmx2ly**' '**musedata2ly**' '**mup2ly**'.

57.2.6 Programmi frontali

Esistono diversi programmi in grado di generare un sorgente LilyPond componendo in modo grafico. Per esempio, è disponibile Denemo,[2] tuttavia, la composizione fatta in questo può essere efficace solo se si studia molto bene l'utilizzo della tastiera, per poter selezionare rapidamente le note da inserire, con le loro caratteristiche.

57.2.7 Progetto Mutopia

Presso *http://www.mutopiaproject.org* è disponibile un archivio di musica di dominio pubblico, realizzato attraverso LilyPond. Da questo archivio è possibile prelevare spartiti o esecuzioni in formato MIDI.

Generalmente conviene prelevare direttamente i file sorgenti LilyPond, per ricompilarli come si vuole. Tuttavia, questi sorgenti sono realizzati nei modi più diversi, ma soprattutto, sono fatti ognuno per una versione particolare di LilyPond.

Una volta prelevati i sorgenti di un'opera a cui si è interessati, si deve provvedere ad allinearne la versione con quella di LilyPond che si intende utilizzare. Il procedimento è semplice se si tratta di un file soltanto, oppure di più file indipendenti, riferiti a porzioni differenti dell'opera. Si propone un esempio con il file *http://www.mutopiaproject.org/ftp/BachJS/BWV777/bach-invention-06/bach-invention-06.ly* :

```
$ convert-ly -e bach-invention-06.ly [Invio]

convert-ly (GNU LilyPond) 1.6.6
Processing 'bach-invention-06.ly' ... Applying conversions:
1.5.33, 1.5.38, 1.5.40, 1.5.49, 1.5.52, 1.5.56, 1.5.58,
1.5.59, 1.5.62, 1.5.67, 1.5.68, 1.5.71, 1.5.72, 1.6.5,
```

In questo caso, la conversione ha funzionato, o almeno dà l'impressione di funzionare, perché il sorgente contiene l'istruzione '**\version**' appropriata:

```
\version "1.3.148"
```

Quando nel sorgente manca l'istruzione '**\version**', bisogna inserirla, dopo essere risaliti a questa indicazione, altrimenti la conversione è impossibile.

Tutto diventa molto difficile quanto il sorgente di un'opera si articola in file differenti. Di solito, chi realizza questa struttura non fornisce indicazioni sul modo corretto di ricompilare il sorgente; d'altro canto, spesso questi file sono generati da altri strumenti. In questi casi, si devono convertire tutti i file, ma spesso, solo il file principale, che richiama gli altri, contiene l'istruzione '**\version**'. Se le cose stanno così, si deve ricopiare quell'istruzione all'inizio di tutti questi file, utilizzando poi '**convert-ly**' con ognuno di questi, sperando di avere successo.

Successivamente si può passare alla compilazione, indicando il file sorgente principale, oppure ognuno dei file sorgenti principali se un'opera è suddivisa in parti separate.

```
$ lilypond --dvi bach-invention-06.ly [Invio]
```

Se lo scopo della compilazione è generare dei file MIDI, si possono incontrare altre difficoltà; la prima, costituita dal fatto che chi ha trascritto l'opera con LilyPond, può averlo fatto in modo da generare spartiti separati per ogni strumento utilizzato: in questo caso, si rischia di ottenere tanti file MIDI quanti sono gli strumenti utilizzati. In questa situazione, bisogna intervenire nel sorgente in modo molto approfondito, per riunire assieme tutti gli spartiti, in un libro solo.

Se la compilazione genera correttamente un file MIDI soltanto, oppure un file per ogni porzione dell'opera, può darsi che gli strumenti usati per l'esecuzione non corrispondano a quelli previsti nell'opera stessa. In generale, il sorgente LilyPond ottenuto dall'archivio del progetto Mutopia contiene un'intestazione con molte informazioni importanti:

```
\header {
    …
 mutopiatitle = "Invention 6"
 mutopiacomposer = "J. S. Bach (1685-1750)"
 mutopiaopus = "BWV 777"
 mutopiainstrument = "Harpsichord, Piano"
    …
}
```

In questo caso, si vede la proposta di utilizzare un clavicembalo o un piano. Data l'epoca a cui si riferisce l'opera trascritta, è probabile

che sia più gradevole il suono di un clavicembalo. Se nel sorgente manca la richiesta esplicita di usare questo strumento, occorre provvedere ad aggiungere l'istruzione appropriata, altrimenti si ottiene il piano che è lo strumento predefinito:

```
\score {
   \context GrandStaff <<
    \context Staff = "one" <<
      \set Staff.instrument = "harpsichord"
      \set Staff.midiInstrument = "harpsichord"
      \voiceone
    >>
    \context Staff = "two" <<
      \set Staff.instrument = "harpsichord"
      \set Staff.midiInstrument = "harpsichord"
      \voicetwo
    >>
  >>
…
```

Naturalmente, la denominazione dello strumento deve corrispondere a quanto previsto da LilyPond (tabella 57.37).

57.3 Riferimenti

- Maurizio Codogno, *Pillole di teoria musicale*, 2002, *http://xmau.com/musica/teoria/pillole.html*
- Ivano Cremonesi, *Info teoria musicale per l'informatico*, 1997, *http://www.newmedia.it/musicopoli/teoria_m.htm*
- *Elementi base di teoria & armonia*, *http://www.soundme.com/theory/teoria.htm*
- *GNU LilyPond*, *http://lilypond.org/*
- *Mutopia project*, *http://www.mutopiaproject.org*

[1] **LilyPond** GNU GPL

[2] **Denemo** GNU GPL

Codici a barre

I codici a barre sono dati memorizzati in forma ottica, attraverso l'uso di barre verticali che possono essere lette e interpretate facilmente con strumenti non troppo complessi. La tecnica dei codici a barre nasce negli anni 1960 e si diffonde negli anni 1970, con lo scopo di identificare rapidamente imballaggi e merci.

Gli standard sui codici a barre sono molti e il problema più importante da risolvere quando si vogliono usare è il decidere quale sia quello più conveniente per i propri fini. Si parla di ***simbologia*** per fare riferimento al tipo di codice a barre, ovvero allo standard di rappresentazione dei dati; la simbologia definisce implicitamente il tipo di dati che possono essere memorizzati. Per fare un esempio abbastanza comune, i codici ISBN dei libri sono scritti usando codici a barre di tipo EAN-13, che si compongono di 13 cifre numeriche; in questo caso, la simbologia è EAN-13, con la quale si vanno a rappresentare in pratica i codici ISBN.

Le rappresentazioni di un codice a barre possono essere a una o a due dimensioni. Nel primo caso è prevista una lettura orizzontale unica, attraversando barre verticali nere e spazi bianchi; queste barre di larghezze differenti e gli spazi di ampiezze diverse sono gli elementi della simbologia; ogni simbologia usa un proprio numero di elementi differenti. I codici a barre a due dimensioni sono rappresentati attraverso dei quadrati o dei rettangoli popolati da celle chiare e celle scure, con cui si rappresenta l'informazione desiderata.

Nel capitolo vengono descritte brevemente alcune simbologie standard di codici a una dimensione; vengono trattati anche alcuni codici a due dimensioni, ma senza entrare nell'analisi della simbologia utilizzata.

58.1 Codice di controllo «modulo *n*»

Alcuni tipi di codici numerici utilizzano un numero di controllo (o codice di controllo) per permettere una verifica facile dell'integrità del dato rappresentato. Esiste un genere comune di algoritmi per il calcolo di tali numeri di controllo, a cui si fa spesso riferimento con la definizione ***modulo n***, dove ***n*** è un numero intero maggiore di uno.

In questo modo si fa riferimento a una somma di valori, ottenuta moltiplicando ogni cifra del numero di partenza, compreso il numero di controllo, per un certo ***peso*** (un altro numero intero positivo), che deve essere un multiplo del numero ***n*** stabilito. Si osservi l'esempio seguente, in cui il numero di controllo finale è calcolato con i pesi dati, dividendo per il modulo 5:

Valore	1	3	5	7	4
Peso	5	4	3	2	1
Peso*Valore	5	12	15	14	4

5 + 12 + 15 + 14 + 4 = 50

50 è divisibile perfettamente per cinque, senza lasciare resti, garantendo l'integrità del valore.

In pratica, il peso che si dà al valore da usare come numero di controllo è solitamente uno, per cui diventa facile il calcolo di questa cifra:

5 + 12 + 15 + 14 = 46;
46 / 5 = 9, con il resto di 1;
5 − 1 = 4, ovvero il valore cercato come numero di controllo.

58.2 UPC-E e UPC-A

UPC è un insieme di simbologie standard utilizzato in particolare negli Stati Uniti e nel Canada. Con UPC-E e UPC-A si possono rappresentare solo cifre numeriche: UPC-A consente di rappresentare 11 cifre più una di controllo; UPC-E consente di rappresentare sei cifre, senza codice di controllo (il controllo di integrità avviene in forma differente).

La figura 58.2 mostra rispettivamente l'esempio di 123456 con UPC-E e di 12345678901 con UPC-A.

Figura 58.2. Esempi di codici a barre con simbologia UPC-E e UPC-A.

Gli elementi utilizzati nella simbologia UPC sono otto, composti da quattro tipi di barre nere e quattro tipi di spazi.

Il codice di controllo di UPC-A si calcola moltiplicando in modo alterno le cifre che compongono il numero, per tre, o per uno, sommando alla fine i risultati. Ciò che si ottiene si divide per 10 e si tiene il resto; infine, 10 meno il resto ottenuto dà il codice di controllo cercato (modulo 10). Per esempio, nel caso di 12345678901 si ottiene il codice di controllo due, in base al calcolo seguente:

Valore	1	2	3	4	5	6	7	8	9	0	1
Peso	3	1	3	1	3	1	3	1	3	1	3
Peso*Valore	3	2	9	4	15	6	21	8	27	0	3

Il totale che si ottiene è 98; pertanto: 98/10 = 9 con un resto di 8; 10−8 = 2.

58.3 EAN-8 e EAN-13

EAN (*European article number*) è un insieme di simbologie standard di origine europea, utilizzato anche in altri paesi, nato come estensione delle simbologie UPC.

EAN-8 e EAN-13 permettono di rappresentare solo cifre numeriche: EAN-8 consente di rappresentare sette cifre più una di controllo, mentre EAN-13 consente di rappresentare 12 cifre numeriche più una di controllo.

La figura 58.4 mostra rispettivamente l'esempio di 1234567 con EAN-8 e di 123456789012 con EAN-13.

Figura 58.4. Esempi di codici a barre con simbologia EAN-8 e EAN-13.

Il codice di controllo di EAN-8 e di EAN-13 si calcola nello stesso modo di UPC-A. Per esempio, nel caso di 1234567 (EAN-8) si ottiene il codice di controllo zero, mentre nel caso di 123456789012 (EAN-13) si ottiene otto.

Valore	1	2	3	4	5	6	7
Peso	3	1	3	1	3	1	3
Peso*Valore	3	2	9	4	15	6	21

Il totale che si ottiene è 60; pertanto: 60/10 = 6 con un resto di 0; 10−0 = 10, da cui si prende solo l'ultima cifra (lo zero).

Valore	1	2	3	4	5	6	7	8	9	0	1	2
Peso	1	3	1	3	1	3	1	3	1	3	1	3
Peso*Valore	1	6	3	12	5	18	7	24	9	0	1	6

Il totale che si ottiene è 92; pertanto: 92/10 = 9 con un resto di 2; 10−2 = 8.

La rappresentazione di ogni cifra numerica può avere tre forme differenti, a cui si attribuisce convenzionalmente una lettera alfabetica maiuscola: A, B e C. Nella simbologia EAN-8, le prime quattro cifre sono rappresentate secondo la forma A, mentre le ultime quattro secondo la forma C. Invece, la rappresentazione delle cifre nella simbologia EAN-13 è più complessa, per l'esigenza di essere compatibile con UPC-A.

Con EAN-13 si rappresentano effettivamente solo le ultime 12 cifre (incluso il codice di controllo), mentre la prima viene determinata in base al modo in cui vengono rappresentate le sei cifre successive. In pratica, a seconda del valore della prima cifra, si determina la combinazione delle forme A e B, per le sei cifre che seguono la prima, mentre le ultime sei sono rappresentate con la forma C.

Valore iniziale						
0	A	A	A	A	A	A
1	A	A	B	A	B	B
2	A	A	B	B	A	B
3	A	A	B	B	B	A
4	A	B	A	A	B	B
5	A	B	B	A	A	B
6	A	B	B	B	A	A
7	A	B	A	B	A	B
8	A	B	A	B	B	A
9	A	B	B	A	B	A

Dallo specchietto si può notare che con una prima cifra pari a zero, le sei cifre successive si rappresentano secondo la forma A, esattamente come avviene nella simbologia UPC-A. In pratica, a parte qualche differenza estetica minima, un codice a barre realizzato con la simbologia EAN-13 è uguale a un altro realizzato con la simbologia UPC-A, quando la prima cifra è pari a zero, tenendo conto che lo zero iniziale non può cambiare nulla anche nel calcolo del codice di controllo finale.

58.4 Codici UCC ed EAN

In generale, la simbologia EAN-13 viene utilizzata per identificare degli «articoli», intesi principalmente come prodotti in vendita. Per poter applicare un codice a barre del genere occorre naturalmente avere ottenuto il numero per uno scopo preciso (l'ente italiano competente per l'attribuzione dei codici EAN è Indicod); tuttavia esiste un prefisso iniziale che fa parte dello standard e serve a stabilire l'o-

rigine del codice. l'origine è un concetto riferito precisamente a chi ha rilasciato il numero, che nella maggior parte dei casi si riferisce a un'organizzazione con competenza nazionale.

Questo prefisso iniziale dipende principalmente da due organizzazioni: UCC (*Uniform code council*) ed EAN (*European article number*). La prima organizzazione, definisce codici in cui la prima cifra identifica l'origine nell'area nell'ambito di Stati Uniti e Canada, mentre la seconda copre tutto il mondo, includendo la codifica UCC a cui si aggiunge la cifra iniziale zero.

La tabella 58.8 mostra in pratica un elenco abbastanza completo dei prefissi EAN, che solitamente si applicano ai codici a barre con simbologia EAN-13. Si può osservare che ormai i codici assegnati all'area Stati Uniti e Canada non hanno più solo lo zero iniziale; resta comunque l'abbinamento tra codice UCC e codice EAN, per cui il primo si trasforma nel secondo aggiungendo uno zero iniziale.

Tabella 58.8. Attribuzione dei prefissi secondo la codifica EAN.

Prefisso	Contesto	ISO 3166	Prefisso	Contesto	ISO 3166
00-13	USA, Canada	US, CA	14-19		
2	uso interno		30-37	Francia	FR
380	Bulgaria	BG	383	Slovenia	SI
384			385	Croazia	HR
386			387	Bosnia Erzegovina	BA
388-389			400-440	Germania	DE
45	Giappone	JP	460-469	Federazione russa	RU
470			471	Taiwan	TW
472-473			474	Estonia	EE
475	Lettonia	LV	476	Azerbaijan	AZ
477	Lituania	LT	478	Uzbekistan	UZ
479	Sri Lanka	LK	480	Filippine	PH
481	Belarus	BY	482	Ucraina	UA
483			484	Moldavia	MD
485	Armenia	AM	486	Georgia	GE
487	Kazakistan	KZ	488		
489	Hong Kong	HK	49	Giappone	JP
50	Gran Bretagna	UK	51		
520	Grecia	GR	521-527		
528	Libano	LB	529	Cipro	CY
530			531	Macedonia	MK
532-534			535	Malta	MT
536-538			539	Irlanda	IE
54	Belgio, Lussemburgo	BE, LU	55		
560	Portogallo	PT	561-568		
569	Islanda	IS	57	Danimarca	DK
58			590	Polonia	PL
591-593			594	Romania	RO
595-598			599	Ungheria	HU
600-601	Sud Africa	ZA	602-608		
609	Mauritius	MU	610		
611	Marocco	MA	612		
613	Algeria	DZ	614-615		
616	Kenia	KE	617-618		
619	Tunisia	TN	620		
621	Siria	SY	622	Egitto	EG
623			624	Libia	LY
625	Giordania	JO	626	Iran	IR
627	Kuwait	KW	628	Arabia Saudita	SA
629	Emirati	AE	63		
64	Finlandia	FI	65-68		
690-693	Cina	CN	694-699		
70	Norwegia	NO	710-728		
729	Israele	IL	73	Svezia	SE
740	Guatemala	GT	741	El Salvador	SV
742	Honduras	HN	743	Nicaragua	NI
744	Costa Rica	CR	745	Panama	PA
746	Repubblica Dominicana	DO	747-749		
750	Messico	MX	751-758		
759	Venezuela	VE	76	Svizzera	CH
770	Colombia	CO	771-772		
773	Uruguay	UY	774		
775	Perù	PE	776		
777	Bolivia	BO	778		
779	Argentina	AR	780	Cile	CL
781-783			784	Paraguay	PY
785			786	Ecuador	EC
787-788			789	Brasile	BR
80-83	Italia	IT	84	Spagna	ES
850	Cuba	CU	851-857		
858	Slovacchia	SK	859	Czech	CZ
860	Jugoslavia	YU	861-866		
867	Nord Korea	KP	868		
869	Turchia	TR	87	Olanda	NL
880	Sud Korea	KR	881-884		
885	Thailandia	TH	886-887		
888	Singapore	SG	889		
890	India	IN	891-892		
893	Vietnam	VN	894-898		
899	Indonesia	ID	90-91	Austria	AT
92			93	Australia	AU
94	Nuova Zelanda	NZ	950-954		
955	Malaysia	MY	956-957		
958	Macau	MO	959-976		
977	periodici (ISSN)		978-979	libri (ISBN)	
980	ricevute di rimborso (*refund receipts*)		981-982	tagliandi di valuta (*common currency coupons*)	
983-989			99	tagliandi (*coupons*)	

Si può osservare in particolare che il prefissi 978 e 979 sono riservati per i libri, mentre il prefisso 977 è riservato per i periodici (riviste). Dal momento che il prefisso EAN definiva originariamente un'area nazionale, è stata attribuita al codice 978 la denominazione ***bookland*** (la terra dei libri). Molta documentazione sui codici a barre e sulla numerazione ISBN fa ancora riferimento a questo nome per indicare tale prefisso.

Un altro prefisso interessante è il due, che rimane libero. Molti negozi usano codici a barre EAN-13, con prefisso due per annotare il prezzo di qualcosa che non è identificabile diversamente, come gli alimentari ottenuti da un banco di salumi e formaggi. La figura 58.9 mostra un esempio reale in cui, tra le altre cose, si annota un prezzo di 11,24 €.

Figura 58.9. Esempio di un codice a barre con simbologia EAN-13, usato per annotare un prezzo.

58.5 ISBN

La simbologia EAN-13 viene usata in diverse situazioni comuni, in particolare per ciò che riguarda le pubblicazioni.

La codifica ISBN (*International standard book number*) permette di identificare un libro in una sua edizione precisa, attraverso un numero composto da 10 cifre. Il numero in questione contiene quattro parti: l'origine (ovvero l'ente che lo ha rilasciato), l'editore, il titolo (il libro) e un numero di controllo finale (una sola cifra).

Per facilitare la lettura di queste informazioni, le quattro parti vengono mostrate solitamente come separate, attraverso un trattino. Per esempio, '**88-8331-223-6**' indica l'origine 88 (Italia), l'editore 8331, il titolo 233 e il numero di controllo 6.

Le varie parti in cui si suddivide l'informazione contenuta in un numero ISBN non sono di lunghezza fissa e la dimensione dell'una limita quella delle altre. In generale sono disponibili i raggruppamenti seguenti.

1. La prima parte identifica l'area (nazionale, linguistica e geografica) e viene attribuita dall'agenzia internazionale dell'ISBN. Il numero assegnato all'Italia è 88.
   - 0-7
   - 80-94
   - 950-994
   - 9950-9989
   - 99900-99999
2. La seconda parte identifica l'editore e viene attribuito dall'agenzia ISBN competente nell'area di riferimento.
   - 00-19
   - 200-699
   - 7000-8499
   - 85000-89999
   - 900000-999999
3. La terza parte identifica il titolo, relativamente all'editore a cui si fa riferimento, utilizzando lo spazio rimanente, escludendo il numero di controllo finale.
4. Il numero di controllo viene calcolato moltiplicando le cifre per un numero che va da 10 a uno, sommando ciò che si ottiene e dividendo per il «modulo 11». In questo modo, il valore che si ottiene può andare da 0 a 10, dove il 10 si rappresenta con una lettera «X».

Per comprendere il meccanismo del calcolo del numero di controllo, conviene vedere un esempio reale, il numero ISBN '**88-8331-223-6**'. In pratica, si tratta di verificare che il numero sei sia effettivamente il numero di controllo corretto.

Valore	8	8	8	3	3	1	2	2	3	6
Peso	10	9	8	7	6	5	4	3	2	1
Peso*Valore	80	72	64	21	18	5	8	6	6	6

Il totale che si ottiene è 286, che diviso per 11 dà esattamente 26, senza resto, a conferma della validità del numero di controllo. Non conoscendo il numero di controllo lo si potrebbe calcolare: la somma che si otterrebbe sarebbe solo 280, che divisa per 11 dà 25 con il resto di 5, per cui, 11 − 5 dà 6, il numero cercato.

Il numero ISBN si trasforma in EAN-13 mettendo anteriormente il valore 978 e togliendo il numero di controllo finale che va sostituito con quello calcolato secondo lo standard EAN-13; naturalmente, i trattini di divisione sono perduti. La figura 58.11 mostra l'esempio del codice ISBN 88-481-0113-5 rappresentato secondo la simbologia EAN-13, che si trasforma nel numero 9788848101134.

Figura 58.11. ISBN 88-481-0113-5.

A seconda del paese in cui si utilizza il numero ISBN, può essere obbligatorio o meno un codice a barre aggiuntivo contenente l'informazione sul prezzo.

58.5.1 ISSN

La codifica ISSN permette di identificare una pubblicazione periodica attraverso una tecnica simile a quella del numero ISBN per i libri. In questo caso, il numero è composto da otto cifre numeriche (compreso il numero di controllo), che vanno inserite in una simbologia EAN-13 aggiungendo all'inizio il prefisso 977, togliendo il numero di controllo ISSN, aggiungendo due cifre che rappresentano il codice del prezzo e aggiungendo il codice di controllo EAN-13. Inoltre, va aggiunto un codice a barre ulteriore per rappresentare due o cinque cifre in cui si indica il numero del fascicolo.

Per esempio, il numero ISSN '**1129-1346**' di una rivista potrebbe tradursi nel codice a barre 9771129134006, con l'aggiunta di 10110, dove il codice del prezzo è assente e si fa riferimento al fascicolo numero 110.

La figura 58.12 mostra l'esempio del codice di due riviste, in cui appare anche un'estensione contenente il numero della pubblicazione.

Figura 58.12. Esempio di due codici ISSN, riferiti a riviste, con un'estensione contenente il numero della pubblicazione (rispettivamente 106 e 23).

58.6 Code 39

La simbologia denominata Code 39, ovvero *3 of 9*, consente di rappresentare 42 simboli, composti da lettere maiuscole, cifre numeriche e pochi altri segni. La tabella 58.13 riepiloga l'insieme di caratteri. Un codice a barre Code 39 è composto da un carattere di inizio, dai dati che deve contenere, da un codice di controllo (che non appare tradotto per la lettura umana) e dal carattere di fine. Il carattere usato per iniziare e per concludere la sequenza è l'asterisco, che quindi non può essere utilizzato nei dati contenuti.

Tabella 58.13. Insieme di caratteri per la simbologia Code 39.

Valore	Carattere	Valore	Carattere
0	0	22	M
1	1	23	N
2	2	24	O
3	3	25	P
4	4	26	Q
5	5	27	R
6	6	28	S
7	7	29	T
8	8	30	U
9	9	31	V
10	A	32	W
11	B	33	X
12	C	34	Y
13	D	35	Z
14	E	36	-
15	F	37	.
16	G	38	space
17	H	39	$
18	I	40	/
19	J	41	+
20	K	42	%
21	L	inizio/fine	*

Figura 58.14. Esempio di codice a barre Code 39.

Gli elementi utilizzati nella simbologia Code 39 sono quattro, composti da due tipi di barre nere e due tipi di spazi. Ogni carattere o segno che può essere rappresentato con questa simbologia, si avvale di cinque barre e di quattro spazi, per un totale di nove elementi. Il nome della simbologia, che si può tradurre come «tre su nove», indica il fatto che tre elementi su nove sono sempre di tipo largo (rispetto agli altri che invece sono stretti).

Il codice di controllo si ottiene sommando assieme i valori abbinati ai segni che si possono rappresentare, secondo la tabella di conversione 58.13, dividendo per 43. Il resto della divisione è il codice di controllo. Nel caso dell'esempio che appare in figura 58.14, si tratta della lettera '**L**':

«CIAO» = 12 + 18 + 10 + 24 = 64

64 / 43 = 1 lasciando un resto di 21 = '**L**'

58.6.1 Code 39 esteso

Esiste anche un'estensione della simbologia Code 39, allo scopo di consentire la rappresentazione dell'ASCII standard completo. Si ottiene questo risultato scrivendo due simboli normali al posto di uno. Naturalmente, dal momento che non esiste un modo per distinguere la codifica standard da quella estesa, è necessario che il lettore di codice a barre sia impostato nel modo più conveniente. La tabella 58.15 mostra l'insieme di caratteri esteso e la corrispondenza con i segni della simbologia Code 39 normale.

Tabella 58.15. Insieme di caratteri esteso per la simbologia Code 39.

ASCII	Code 39	ASCII	Code 39	ASCII	Code 39	ASCII	Code 39
<NUL>	%U	*<SP>*	=	@	%V	‘	%W
<SOH>	$A	!	/A	A	A	a	+A
<STX>	$B	"	/B	B	B	b	+B
<ETX>	$C	#	/C	C	C	c	+C
<EOT>	$D	$	/D	D	D	d	+D
<ENQ>	$E	%	/E	E	E	e	+E
<ACK>	$F	&	/F	F	F	f	+F
<BEL>	$G	’	/G	G	G	g	+G
<BS>	$H	(	/H	H	H	h	+H
<HT>	$I	)	/I	I	I	i	+I
<LF>	$J	*	/J	J	J	j	+J
<VT>	$K	+	/K	K	K	k	+K
<FF>	$L	,	/L	L	L	l	+L
<CR>	$M	-	/M	M	M	m	+M
<SO>	$N	.	/N	N	N	n	+N
<SI>	$O	/	/O	O	O	o	+O
<DLE>	$P	0	0	P	P	p	+P
<DC1>	$Q	1	1	Q	Q	q	+Q
<DC2>	$R	2	2	R	R	r	+R
<DC3>	$S	3	3	S	S	s	+S
<DC4>	$T	4	4	T	T	t	+T
<NAK>	$U	5	5	U	U	u	+U
<SYN>	$V	6	6	V	V	v	+V
<ETB>	$W	7	7	W	W	w	+W
<CAN>	$X	8	8	X	X	x	+X
<EM>	$Y	9	9	Y	Y	y	+Y
<SUB>	$Z	:	/Z	Z	Z	z	+Z
<ESC>	%A	;	%F	[	%K	{	%P
<FS>	%B	<	%G	\	%L	|	%Q
<GS>	%C	=	%H	]	%M	}	%R

ASCII	Code 39	ASCII	Code 39	ASCII	Code 39	ASCII	Code 39
<RS>	%D	>	%I	^	%N	~	%S
<US>	%E	?	%J	–	%O	*<DEL>*	%T

58.7 Code 128

La simbologia denominata Code 128, ovvero USS Code 128, consente di rappresentare 106 simboli, che possono essere interpretati secondo tre insiemi di caratteri, distinti in base a una lettera: A, B e C. Per questa ragione, un codice a barre realizzato secondo la simbologia Code 128 inizia sempre con la dichiarazione dell'insieme di caratteri. L'informazione non ha una lunghezza predeterminata e può essere modificato l'insieme di caratteri quando serve. La simbologia Code 128 prevede l'inserimento di un codice di controllo, calcolato automaticamente per consentire la verifica della lettura meccanica, ma questa cifra non viene mostrata per la lettura umana.

Tabella 58.16. Insiemi di caratteri per la simbologia Code 128. L'ultima colonna riporta anche la descrizione della rappresentazione a barre, dove un numero da uno a tre indica la larghezza della barra, '**b**', o dello spazio, '**s**', dalla grandezza più piccola alla più grande. Si osservi che lo stop ha una barra in più alla fine.

Valore	Code A	Code B	Code C	Barre e spazi: b s b s b s
0	*<SP>*	*<SP>*	00	2 1 2 2 2 2
1	!	!	01	2 2 2 1 2 2
2	"	"	02	2 2 2 2 2 1
3	#	#	03	1 2 1 2 2 3
4	$	$	04	1 2 1 3 2 2
5	%	%	05	1 3 1 2 2 2
6	&	&	06	1 2 2 2 1 3
7	’	’	07	1 2 2 3 1 2
8	(	(	08	1 3 2 2 1 2
9	)	)	09	2 2 1 2 1 3
10	*	*	10	2 2 1 3 1 2
11	+	+	11	2 3 1 2 1 2
12	,	,	12	1 1 2 2 3 2
13	-	-	13	1 2 2 1 3 2
14	.	.	14	1 2 2 2 3 1
15	/	/	15	1 1 3 2 2 2
16	0	0	16	1 2 3 1 2 2
17	1	1	17	1 2 3 2 2 1
18	2	2	18	2 2 3 2 1 1
19	3	3	19	2 2 1 1 3 2
20	4	4	20	2 2 1 2 3 1
21	5	5	21	2 1 3 2 1 2
22	6	6	22	2 2 3 1 1 2
23	7	7	23	3 1 2 1 3 1
24	8	8	24	3 1 1 2 2 2
25	9	9	25	3 2 1 1 2 2
26	:	:	26	3 2 1 2 2 1
27	;	;	27	3 1 2 2 1 2
28	<	<	28	3 2 2 1 1 2
29	=	=	29	3 2 2 2 1 1
30	>	>	30	2 1 2 1 2 3
31	?	?	31	2 1 2 3 2 1
32	@	@	32	2 3 2 1 2 1
33	A	A	33	1 1 1 3 2 3
34	B	B	34	1 3 1 1 2 3
35	C	C	35	1 3 1 3 2 1
36	D	D	36	1 1 2 3 1 3
37	E	E	37	1 3 2 1 1 3
38	F	F	38	1 3 2 3 1 1
39	G	G	39	2 1 1 3 1 3
40	H	H	40	2 3 1 1 1 3
41	I	I	41	2 3 1 3 1 1
42	J	J	42	1 1 2 1 3 3
43	K	K	43	1 1 2 3 3 1
44	L	L	44	1 3 2 1 3 1
45	M	M	45	1 1 3 1 2 3

Valore	Code A	Code B	Code C	Barre e spazi: b s b s b s
46	N	N	46	1 1 3 3 2 1
47	O	O	47	1 3 3 1 2 1
48	P	P	48	3 1 3 1 2 1
49	Q	Q	49	2 1 1 3 3 1
50	R	R	50	2 3 1 1 3 1
51	S	S	51	2 1 3 1 1 3
52	T	T	52	2 1 3 3 1 1
53	U	U	53	2 1 3 1 3 1
54	V	V	54	3 1 1 1 2 3
55	W	W	55	3 1 1 3 2 1
56	X	X	56	3 3 1 1 2 1
57	Y	Y	57	3 1 2 1 1 3
58	Z	Z	58	3 1 2 3 1 1
59	[	[	59	3 3 2 1 1 1
60	\	\	60	3 1 4 1 1 1
61	]	]	61	2 2 1 4 1 1
62	^	^	62	4 3 1 1 1 1
63	–	–	63	1 1 1 2 2 4
64	*<NUL>*	‘	64	1 1 1 4 2 2
65	*<SOH>*	a	65	1 2 1 1 2 4
66	*<STX>*	b	66	1 2 1 4 2 1
67	*<ETX>*	c	67	1 4 1 1 2 2
68	*<EOT>*	d	68	1 4 1 2 2 1
69	*<ENQ>*	e	69	1 1 2 2 1 4
70	*<ACK>*	f	70	1 1 2 4 1 2
71	*<BEL>*	g	71	1 2 2 1 1 4
72	*<BS>*	h	72	1 2 2 4 1 1
73	*<HT>*	i	73	1 4 2 1 1 2
74	*<LF>*	j	74	1 4 2 2 1 1
75	*<VT>*	k	75	2 4 1 2 1 1
76	*<FF>*	l	76	2 2 1 1 1 4
77	*<CR>*	m	77	4 1 3 1 1 1
78	*<SO>*	n	78	2 4 1 1 1 2
79	*<SI>*	o	79	1 3 4 1 1 1
80	*<DLE>*	p	80	1 1 1 2 4 2
81	*<DC1>*	q	81	1 2 1 1 4 2
82	*<DC2>*	r	82	1 2 1 2 4 1
83	*<DC3>*	s	83	1 1 4 2 1 2
84	*<DC4>*	t	84	1 2 4 1 1 2
85	*<NAK>*	u	85	1 2 4 2 1 1
86	*<SYN>*	v	86	4 1 1 2 1 2
87	*<ETB>*	w	87	4 2 1 1 1 2
88	*<CAN>*	x	88	4 2 1 2 1 1
89	*<EM>*	y	89	2 1 2 1 4 1
90	*<SUB>*	z	90	2 1 4 1 2 1
91	*<ESC>*	{	91	4 1 2 1 2 1
92	*<FS>*	|	92	1 1 1 1 4 3
93	*<GS>*	}	93	1 1 1 3 4 1
94	*<RS>*	~	94	1 3 1 1 4 1
95	*<US>*	*<DEL>*	95	1 1 4 1 1 3
96	FNC 3	FNC 3	96	1 1 4 3 1 1
97	FNC 2	FNC 2	97	4 1 1 1 1 3
98	Shift	Shift	98	4 1 1 3 1 1
99	Code C	Code C	99	1 1 3 1 4 1
100	Code B	FNC 4	Code B	1 1 4 1 3 1
101	FNC 4	Code A	Code A	3 1 1 1 4 1
102	FNC 1	FNC 1	FNC 1	4 1 1 1 3 1
103	Start A	Start A	Start A	2 1 1 4 1 2
104	Start B	Start B	Start B	2 1 1 2 1 4
105	Start C	Start C	Start C	2 1 1 2 3 2
	Stop	Stop	Stop	2 3 3 1 1 1 2

Gli insiemi di caratteri A, B e C sono elencati nella tabella 58.16. Come si può vedere, l’insieme A consente di rappresentare l’alfabeto maiuscolo, le cifre numeriche, vari simboli di punteggiatura e caratteri di controllo; in pratica, ciò che si può rappresentare con la prima parte della codifica ASCII, tenendo conto che le posizioni non corrispondono. L’insieme B consente di rappresentare praticamente tutto l’alfabeto ASCII, compresa la distinzione tra lettere maiuscole e minuscole, a esclusione dei caratteri di controllo. L’insieme C consente di rappresentare valori numerici a coppie, per cui, un valore composto da un numero dispari di cifre acquisisce uno zero iniziale.

Figura 58.17. Esempio di due codici Code 128, realizzati usando l’insieme B e l’insieme C, per rappresentare due dati equivalenti: «Ciao» e 35736579.

La figura 58.17 mostra la comparazione tra un codice a barre realizzato usando l’insieme B e un altro con l’insieme C. L’informazione contenuta è la stessa dal punto di vista dei valori rappresentabili (eventualmente si osservino le tabelle degli insiemi di caratteri). Se fosse possibile sovrapporre i due codici, si noterebbe che cambia solo la parte iniziale, quella in cui si dichiara l’insieme di caratteri, e la parte finale, dove si inserisce il codice di controllo. Si può anche osservare che gli ultimi elementi rimangono uguali e sono quelli che contengono il codice di conclusione.

Volendo entrare nel dettaglio, l’esempio di figura 58.17 si compone di: ‘**`Start B`**’, 35, 73, 65, 79, ***codice_di_controllo***, ‘**`Stop`**’; ovvero, nel secondo caso, ‘**`Start C`**’, 35, 73, 65, 79, ***codice_di_controllo***, ‘**`Stop`**’.

Quando si vuole cambiare l’insieme di caratteri, si inserisce un simbolo ‘**`Code A`**’, ‘**`Code B`**’ o ‘**`Code C`**’, per introdurre l’insieme a cui si vuole fare riferimento da quel punto in poi.

Il codice di controllo finale si calcola sommando il valore corrispondente alla dichiarazione iniziale dell’insieme di caratteri alla somma dei valori successivi, moltiplicati per la loro posizione. La somma complessiva va divisa per 103 e il resto che si ottiene è il codice di controllo finale. Si osservi l’esempio:

	Start B	C	i	a	o	Stop
Peso	1	1	2	3	4	
Valore	104	35	73	65	79	
Peso*Valore	104	35	146	195	316	

Pertanto:

104 + 35 + 146 + 195 + 316 = 796

796 / 103 = 7 con resto di 75. Il codice di controllo è il simbolo corrispondente al valore 75.

Usando l’insieme di caratteri C la cosa cambia, perché il simbolo iniziale ha il valore 105, per cui il codice di controllo finale è 76:

	Start C	35	73	65	79	Stop
Peso	1	1	2	3	4	
Valore	105	35	73	65	79	
Peso*Valore	105	35	146	195	316	

Dalla simbologia Code 128 derivano diverse applicazioni speciali che si distinguono per avere una struttura particolare. Esiste anche una variante che consente la rappresentazione di più caratteri rispetto all’ASCII standard.

58.7.1 UCC/EAN 128

La simbologia UCC/EAN 128 è un’applicazione di Code 128, in cui si inizia dichiarando l’insieme di caratteri e si inserisce subito dopo il simbolo ‘**`FNC 1`**’ (*Function code one*).[1]

Dopo il simbolo ‘**`FNC 1`**’ viene indicato un codice che indica il tipo di applicazione. Si fa riferimento a questo con la sigla AI (*Application identifier*). L’informazione in questione viene rappresentata per la lettura umana tra parentesi tonde, proprio per facilitare l’interpretazione, a sottolineare il fatto che si tratta di un prefisso. Naturalmente, l’informazione può essere anche più complessa e altre parti dei

dati successivi possono essere separate ed evidenziate nello stesso modo o in modi differenti.

58.8 ITF, ovvero i25

La simbologia ITF, o i25, nota come Interleaved two of five, ovvero «interfogliata due su cinque», consente la rappresentazione di soli valori numerici, di lunghezza indefinita, purché in numero di cifre pari. La simbologia prevede una cifra di controllo finale, opzionale.

La simbologia si compone di due tipi di barre e due tipi di spazi. Ogni simbolo si rappresenta con cinque barre e cinque spazi, dove due barre su cinque e due spazi su cinque sono più larghi. La simbologia è detta «interfogliata», perché una cifra è codificata nelle barre e la successiva lo è negli spazi che separano gli elementi del carattere precedente. Prima dei dati è previsto un simbolo iniziale, composto da una barra sottile, uno spazio breve, una barra sottile e un altro spazio breve; alla fine c'è un simbolo finale, composto da una barra larga, uno spazio sottile e una barra sottile.

Figura 58.20. Esempio di codice a barre con simbologia interfogliata due su cinque.

58.9 Barcode

Barcode[2] è una libreria e anche un programma molto semplice per la realizzazione di codici a barre standard a una sola dimensione. Il codice a barre viene realizzato in forma di file PostScript, che successivamente può essere utilizzato direttamente o convertito in altri formati grafici, attraverso programmi standard.

Barcode, come programma, si compone in pratica dell'eseguibile '**`barcode`**'. Questo genera normalmente un file PostScript che, salvo l'utilizzo dell'opzione '**`-o`**', viene emesso attraverso lo standard output.

```
barcode [opzioni]
```

Teoricamente si può usare Barcode anche senza l'indicazione di alcun argomento, per ottenere la conversione automatica di quanto inserito attraverso lo standard input, scegliendo la simbologia in modo automatico. Tuttavia, in genere è opportuno accertarsi di selezionare la simbologia attraverso l'opzione '**`-e`**'.

Le opzioni che vengono mostrate qui sono poche; in particolare mancano quelle che servono a definire la stampa di codici a barre in modo organizzato sul foglio, allo scopo di centrare delle etichette adesive. Per approfondire basta leggere la documentazione originale: *info barcode*.

Tabella 58.21. Alcune opzioni.

Opzione	Descrizione
`-e` ***simbologia***	Consente di specificare la simbologia dei codici a barre che si vogliono realizzare (*encoding*). Sono disponibili diverse parole chiave da usare come argomento dell'opzione, secondo la tabella 58.22.
`-i` ***file***	Stabilisce il nome di un file contenente un elenco di codici da trasformare in codici a barre. Se manca questa indicazione e se manca anche l'opzione '**`-b`**', viene letto lo standard input. La lettera dell'opzione richiama mnemonicamente la parola «input».

Opzione	Descrizione
`-b` ***stringa***	Consente di indicare una stringa da convertire in un codice a barre singolo. La stringa deve essere compatibile con i tipi di dati che possono essere rappresentati con la simbologia prescelta. La lettera dell'opzione richiama mnemonicamente la parola *barcode*, ovvero il codice a barre che si vuole ottenere dalla stringa.
`-E`	Genera un file PostScript incapsulato (EPS).

Segue la descrizione di alcuni esempi.

- `$ barcode -E -e i25 -b "12345678" > prova.ps` [*Invio*]

  Converte il numero 12345678 in un codice a barre secondo la simbologia interfogliata due su cinque, generando il file '`prova.ps`', di tipo EPS.

- `$ barcode -E -e i25 -b "12345678" -o prova.ps` [*Invio*]

  Esattamente come nell'esempio precedente.

- `$ barcode -E -e isbn -b "88-386-4177-3" -o prova.ps` [*Invio*]

  Come nell'esempio precedente, usando la simbologia EAN-13 per rappresentare un codice ISBN, come indicato con l'opzione '**`-b`**'.

Generalmente, Barcode viene utilizzato indicando il tipo di simbologia, attraverso una parola chiave secondo l'elenco della tabella 58.22. In generale, si tratta di nomi che fanno riferimento al tipo di codice a barre; in alcuni casi, il nome indica anche altre caratteristiche. Per esempio, la parola chiave '**`code128b`**' indica la simbologia Code 128, nella quale si utilizza l'insieme di caratteri B; inoltre, la parola chiave '**`isbn`**' fa riferimento al codice ISBN, che viene rappresentato correttamente secondo la simbologia EAN-13.

Tabella 58.22. Simbologie disponibili con Barcode.

Simbologia	Parola chiave
UPC-A	upc-a
UPC-E	upc-e
UPC-A o UPC-E automaticamente	upc
EAN-8	ean8, ean-8
EAN-13	ean13, ean-13
EAN-8 o EAN-13 automaticamente	ean
Code 39	code39, 39
Code 128 insieme di caratteri B	code128b, 128b
Code 128 insieme di caratteri C	code128c, 128c
Code 128 insieme di caratteri A, B o C, automaticamente	code128, 128
Code 128 libero	128raw
interfogliata due su cinque	i25, interleaved 2 of 5
Codabar	cdr, codabar
Plessey	pls, plessey
ISBN attraverso EAN-13	isbn

Nel caso dell'uso della simbologia Code 128, si pone il problema di rappresentare i codici da <*NUL*> a <*US*>, il codice <*DEL*> e le funzioni speciali previste dalla simbologia. Usando l'eseguibile '**`barcode`**' diventa difficile indicare questi simboli; con la libreria non ci sono problemi a indicare i codici ASCII da <*SOH*> a <*US*> e il codice <*DEL*>, mentre negli altri casi viene fatta una trasformazione, come rappresentato nella tabella 58.23.

Tabella 58.23. Rappresentazione dei caratteri speciali per la simbologia Code 128, quando si usa la libreria Barcode.

Codice o simbolo	Ottale	Decimale	Esadecimale
<*NUL*>	200_8	128	80_{16}
FNC 1	301_8	193	$C1_{16}$
FNC 2	302_8	194	$C2_{16}$
FNC 3	303_8	195	$C3_{16}$
FNC 4	304_8	196	$C4_{16}$

Sempre a proposito della simbologia Code 128, è possibile utilizzando la parola chiave '**128raw**' per indicare un codice attraverso una stringa espressa nella forma:

```
n_1 n_2… n_n
```

In pratica, si indicano dei numeri separati tra loro da uno spazio. I numeri in questione rappresentano il valore indicato nella tabella 58.16; inoltre occorre tenere in considerazione che va fornito anche il valore iniziale, con il quale si definisce solitamente l'insieme di caratteri A, B o C, mentre il codice di controllo finale e lo stop sono aggiunti automaticamente.

58.10 Codice QR

Il codice a barre QR (*quick response*), noto anche con la sigla *mobtag*, è di tipo bidimensionale, costituito in pratica da una matrice di punti, disposti su una superficie quadrata. Questo tipo di codice a barre viene usato per contenere informazioni di vario tipo, sia numeriche, sia testuali; spesso si usa per indicare indirizzi URL da leggere agevolmente attraverso la telecamera di un telefono mobile.

Figura 58.24. Codice QR contenente il riferimento alla pagina *http://informaticalibera.net* , in quattro diverse alternative.

Il codice QR utilizza una codifica propria dei dati che varia in funzione del tipo di dati da rappresentare. In linea di massima, si può considerare che la capacità massima di questo sistema è di 2953 byte.

Il codice QR permette di utilizzare quattro livelli diversi di ridondanza, utili per la correzione degli errori di lettura. Si utilizzano per questo le lettere L, M, Q e H, per indicare livelli crescenti di ridondanza. Logicamente, un codice QR con livello di ridondanza L ha una capacità maggiore rispetto a quello con livello H.

58.11 ZBar

ZBar[3] è una libreria per la lettura di vari tipi di codici a barre, inclusi i codici QR, accompagnata generalmente da un programma di esempio per il suo utilizzo. La libreria ha lo scopo di interpretare un'immagine digitale che contenga un codice a barre.

Il pacchetto prodotto dagli sviluppatori della libreria ZBar contiene anche due programmi di esempio: '**zbarcam**' per l'acquisizione dei codici a barre attraverso una fotocamera, e '**zbarimg**' per l'interpretazione di un'immagine già acquisita. L'uso di '**zbarimg**' è molto semplice:

```
zbarimg [opzioni] file
```

Supponendo che il file '`codice.png`' contenga un codice QR contenente un indirizzo URL, ecco come si potrebbe usare il programma:

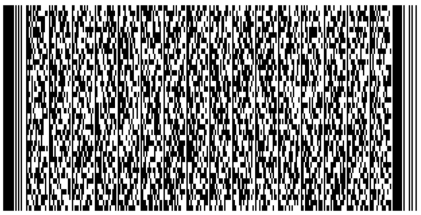
```
$ zbarimg codice.png [Invio]

QR-Code:http://it.m.wikipedia.org/wiki/Pagina_principale
scanned 1 barcode symbols from 1 images in 0.01 seconds
```

58.12 Codice PDF417

Il codice a barre PDF417 (*portable data file*) è di tipo bidimensionale e consente la rappresentazione di testo secondo la codifica ASCII, escludendo però molti codici privi di rappresentazione tipografica.

Figura 58.26. Codice PDF417 contenente il riferimento alla pagina *http://informaticalibera.net* .



La capacità di un codice a barre PDF417 è di circa un migliaio di caratteri; tuttavia va valutato il livello di correzione di errore, da zero a otto, che può ridurre questo spazio.

58.13 Codice DataMatrix e Semacode

Il codice a barre DataMatrix è di tipo bidimensionale e consente la rappresentazione di testo, secondo varie codifiche.

Figura 58.27. Codice DataMatrix contenente il riferimento alla pagina *http://informaticalibera.net* . In tal caso equivale a un codice Semacode.

La capacità di un codice a barre DataMatrix è di circa due migliaia di caratteri e il sistema di correzione degli errori non prevede la scelta di livelli regolabili.

Il codice Semacode è in pratica un codice DataMatrix utilizzato per rappresentare indirizzi URL.

58.14 Riferimenti

- *GS1, the global language of business*, *http://www.gs1.org/*
- Tino Hempel, *Die Europäische Artikelnummer (EAN)*, *http://www.tinohempel.de/info/mathe/ean/ean.htm*
- *Barcoding for Beginners & Bar Code FAQ*, *http://www.idautomation.com/barcoding4beginners.html*
- *Code 39 Barcode FAQ and Tutorial*, *http://www.idautomation.com/code39fa39q.html*
- *Code 128 / USS Code-128 Barcode FAQ & Tutorial*, *http://www.idautomation.com/code128fa28q.html*
- *PDF417 Barcode FAQ*, *http://www.idautomation.com/pdf417fa17q.html*
- *UPC / EAN Barcode Font Data Sheet*, *http://www.bizfonts.com/upc-ean/faq.htm*
- *PDF417*, *http://www.pdf417.com/*
- Russ Adams, *Barcode 1*, *http://www.barcode-1.com/*
- *International ISBN Agency: Coordinator of the International Standard Book Number system*, *http://www.isbn-international.org*
- International ISBN Agency, *The ISBN Users' Manual*, *http://www.isbn.org/standards/home/isbn/International/ISBNmanual.asp*
- Wikipedia, *Codice QR*, *http://it.wikipedia.org/wiki/Codice_QR*
- ZXing Project, *QR code generator*, *http://zxing.appspot.com/generator*
- Wikipedia, *Data Matrix*, *http://it.wikipedia.org/wiki/Data_Matrix*

- Raco industries, *Free 1D and 2D barcode generator*, *http://www.racoindustries.com/barcodegenerator*
- Inlite, *Clearimage free online barcode decoder*, *http://online-barcode-reader.inliteresearch.com*

[1] L'uso comune è dell'insieme di caratteri C, ma ciò non è imposto dallo standard.

[2] **Barcode** GNU GPL

[3] **ZBar** GNU LGPL

Usare i programmi di scrittura visuale

Nelle sezioni successive viene descritto l'uso generale dei programmi di scrittura di tipo visuale, mostrando qualche esempio secondo la configurazione locale italiana.

59.1 La pagina

La configurazione della pagina consente principalmente di definire il formato di carta, l'orientamento, il numero di colonne, i margini, le righe di intestazione e di fondo pagina.

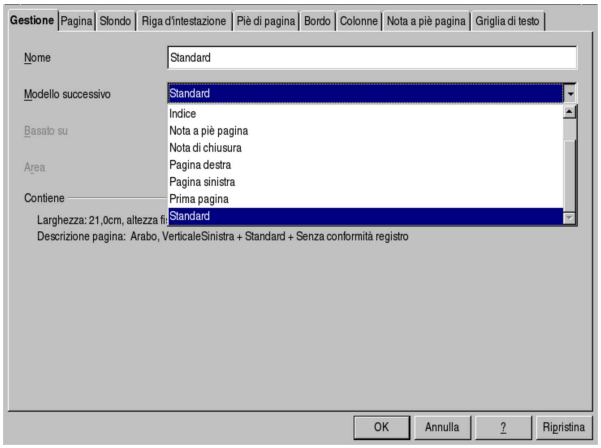
59.1.1 Pagine non omogenee

I programmi di scrittura più sofisticati, consentono di realizzare un documento le cui pagine possono avere caratteristiche differenti tra di loro. Per esempio, ci potrebbe essere una prima pagina in formato A3, mentre le pagine successive sono in formato A4. Questa facoltà potrebbe essere utile per organizzare in modo differente le pagine della copertina rispetto al corpo centrale, pur mantenendo il formato della carta uguale, come per qualunque altro scopo.

Purtroppo, il controllo di queste funzioni, con l'ausilio di programmi visuali, diventa molto difficile e in generale è sconsigliabile di servirsene. In altri termini, è preferibile la gestione di uno stile di pagina uniforme, piuttosto di mettersi nei guai.

Quando si vogliono sfruttare caratteristiche «particolari» di un programma di scrittura visuale, si crea una dipendenza con lo stesso. Di conseguenza, il passaggio a un altro programma simile, anche se con funzionalità sostanzialmente equivalenti, diventa traumatico. Pertanto, dove possibile, conviene sempre limitarsi a utilizzare funzionalità comuni.

Figura 59.1. LibreOffice Writer: *Formato*, *Pagina*, *Gestione*. Attraverso questa scheda è possibile assegnare a ogni pagina una gestione differente, a cui corrisponde una configurazione propria. In generale, è meglio limitarsi alla gestione «standard».



59.1.2 Formato della carta, orientamento e margini esterni

La prima cosa da fare, quando si vuole scrivere un documento, è quello di stabilire il formato della carta che viene usato per stamparlo e i suoi margini esterni. Con LibreOffice Writer, si accede alla configurazione di tutte le opzioni che riguardano la pagina, attraverso la voce *Pagina* del menù *Formato*; con AbiWord, le funzioni specifiche per quello che serve qui si trovano nella voce *Imposta pagina* del menù *File*.

Figura 59.2. Identificazione dei margini. Di norma, quando si ruota il foglio, i margini non seguono la rotazione.

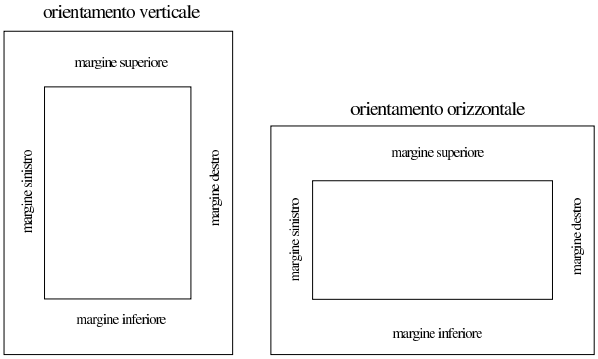
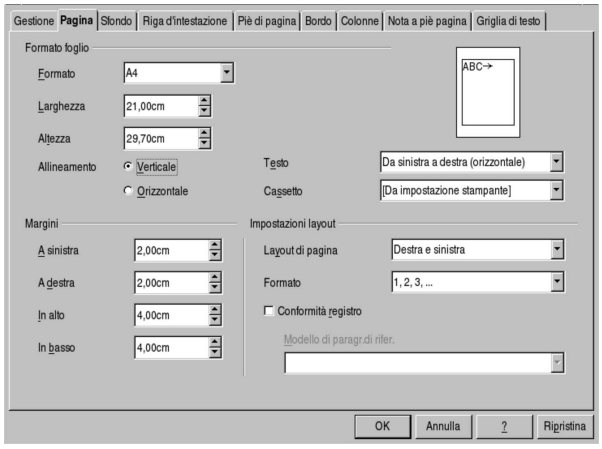


Figura 59.3. AbiWord: *File*, *Imposta pagina*.

Figura 59.4. LibreOffice Writer: *Formato*, *Pagina*, *Pagina*.



59.1.3 Intestazione e fondo pagina

I programmi gestiscono in modo vario la collocazione delle righe di intestazione e a fondo pagina. Queste righe si inseriscono nello spazio del margine superiore e inferiore, rispettivamente; tuttavia, in certi casi, la loro collocazione è indipendente dal margine già stabilito, in altri lo aumenta implicitamente.

Figura 59.5. Collocazione dell'intestazione e del fondo pagina secondo AbiWord.

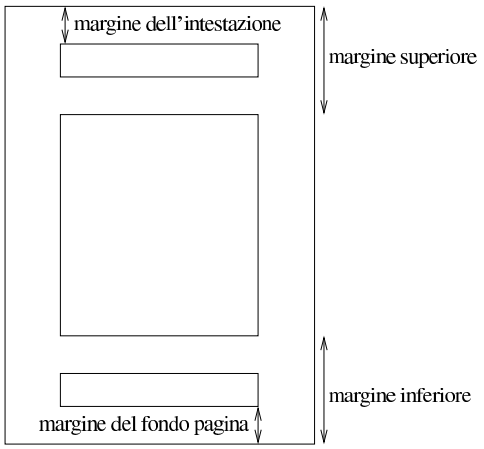
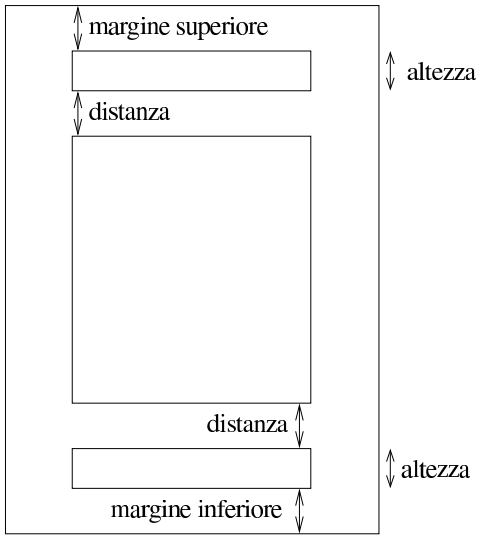


Figura 59.6. Collocazione dell'intestazione e del fondo pagina secondo LibreOffice Writer.



Generalmente, occorre attivare la presenza di intestazione e fondo pagina, quindi si può intervenire per regolarne la posizione. Con AbiWord si attiva l'intestazione con la selezione di *Inserisci, Intestazione*; si attiva il fondo pagina con *Inserisci, Piè pagina.* Con LibreOffice Writer si deve accedere alla scheda *Formato, Pagina, Riga d'intestazione*, oppure *Formato, Pagina, Piè di pagina.*

Figura 59.7. LibreOffice Writer: *Formato, Pagina, Riga d'intestazione*, con la casella di attivazione già selezionata. Si può osservare anche l'indicazione di distanza e altezza.

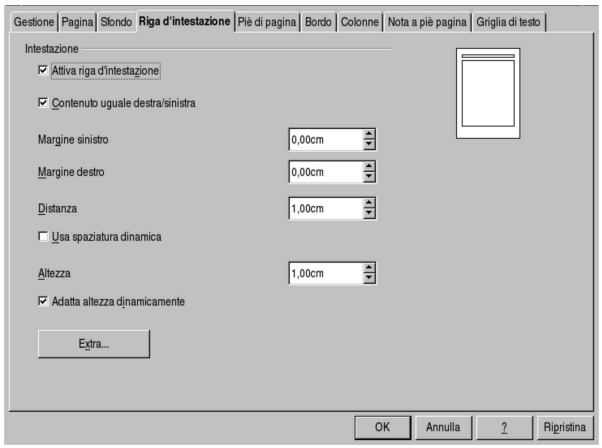


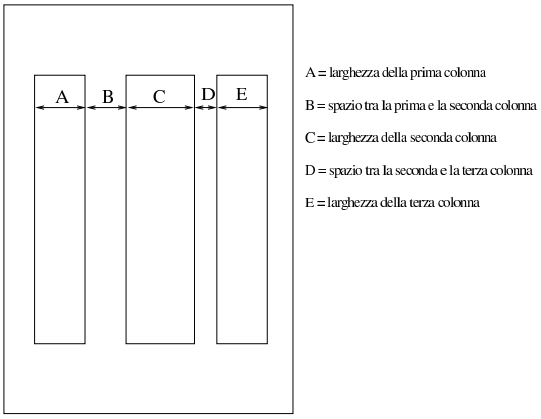
Figura 59.8. AbiWord: *File, Imposta pagina, Margini*. Si può osservare che assieme ai margini esterni si indicano anche i margini di intestazione e fondo pagina.

Il testo contenuto nell'intestazione e nel fondo pagina, viene suddiviso solitamente in tre parti, allineate, rispettivamente: a sinistra, al centro, a destra. Per distinguere le tre porzioni dell'intestazione, si usano normalmente degli stop di tabulazione; in pratica, si scrive la porzione che va collocata a sinistra, quindi si inserisce un carattere di tabulazione con il tasto [*Tab*], si scrive il testo da centrare, si inserisce un altro carattere di tabulazione e si conclude con il testo allineato a destra. Evidentemente, se il proprio programma di scrittura non è già predisposto in questo modo, ammesso che consenta di gestire gli stop di tabulazione, dovrebbe essere possibile crearne uno, centrato, collocato alla metà della larghezza dell'intestazione (o in qualunque altra posizione si preferisca), e un altro allineato a destra, sul bordo destro.

59.1.4 Colonne

Il testo di un documento si svolge normalmente in una sola colonna, nell'ambito dei margini esterni del foglio. Il programma di scrittura visuale comune offre la possibilità di scrivere usando più colonne, a volte consentendo di regolare l'ampiezza delle stesse, lo spazio tra di loro ed eventualmente la presenza di un filo di separazione.

Figura 59.9. Variabili da considerare nella suddivisione in più colonne del testo. Oltre a quello che si può intuire dall'immagine, spesso è possibile inserire delle linee di separazione tra le colonne; potrebbe essere consentita la bordatura; potrebbe essere indicato espressamente il flusso del testo; potrebbe essere specificata anche l'altezza delle colonne, indipendentemente dai margini esterni.



Con AbiWord si configura la suddivisione in più colonne attraverso la selezione di *Formato*, *Colonne*; con LibreOffice Writer si accede a questa funzione con *Formato*, *Colonne*, oppure anche con *Formato*, *Pagina*, *Pagina*, *Colonne*.

Figura 59.10. AbiWord: *Formato*, *Colonne*, con la casella di selezionata della linea separatrice attiva.

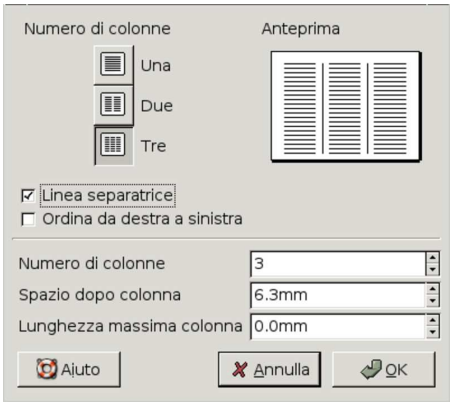
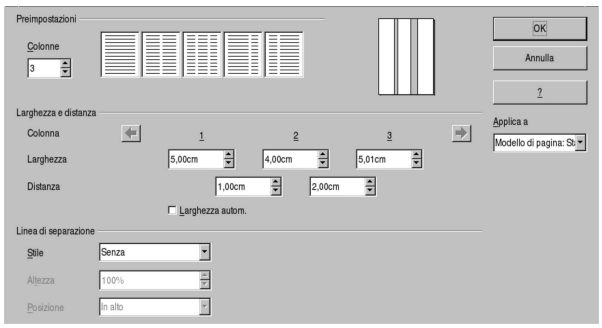


Figura 59.11. LibreOffice Writer: *Formato, Pagina, Colonne*. In questo caso, la distanza tra le colonne è uniforme e vi viene inserito un filo sottile.

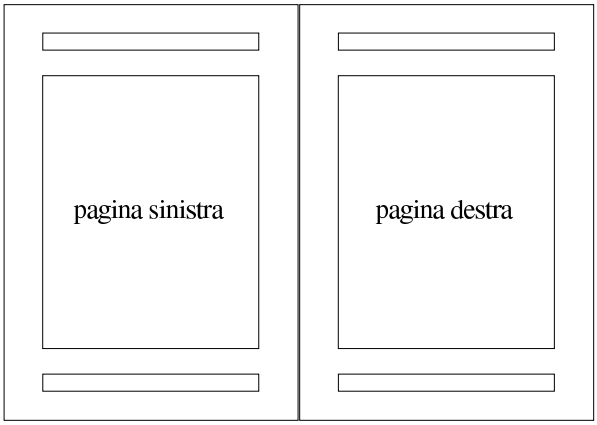
Figura 59.12. LibreOffice Writer: *Formato, Pagina, Colonne*. In questo caso, la distanza tra le colonne è indipendente.



59.1.5 Pagine destre e sinistre

Uno dei problemi che si affronta nella scrittura di un documento importante, è quello di distinguere l'impaginazione delle pagine destre rispetto a quelle sinistre (ovvero delle pagine dispari rispetto a quelle pari). In generale, con un programma di scrittura visuale, se si vuole gestire una qualche differenza, conviene limitarsi alla riga di intestazione e a quella di fondo pagina. In pratica, si tratta di definire che tali informazioni sono distinte tra pagine destre e sinistre, per poter inserire testi differenti tra i due casi.

Figura 59.13. Quando si apre un libro, le due pagine che si vedono sono, rispettivamente, la pagina sinistra e la pagina destra. Generalmente, si considera la prima di copertina come una pagina destra.



Con AbiWord si configurano le particolarità di intestazione e fondo pagina selezionando *Formato*, *Formatta intestazioni/piè di pagina*; con LibreOffice Writer si accede a queste funzioni con *Formato*, *Pagina*, *Riga d'intestazione* o *Piè di pagina*.

Figura 59.14. AbiWord: *Formato, Formatta intestazioni/piè di pagina*. In questo caso, sono selezionate solo le caselle necessarie a gestire pagine destre e sinistre.

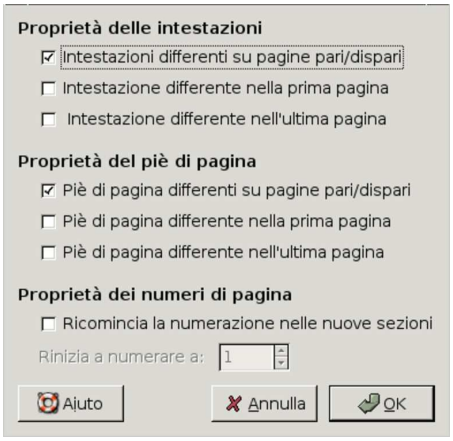
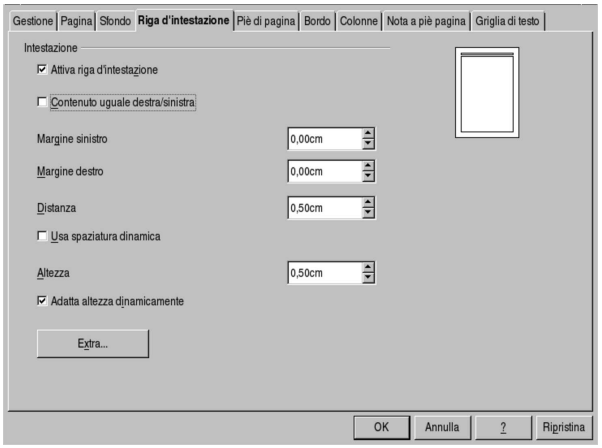


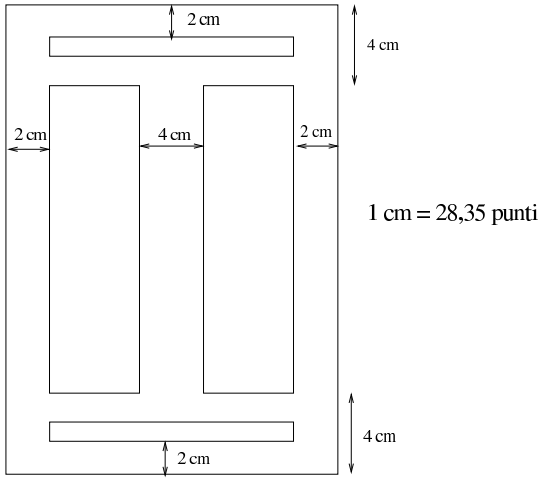
Figura 59.15. LibreOffice Writer: *Formato, Pagina, Riga d'intestazione*. In questo caso, è attiva la riga di intestazione e si distingue tra le pagine destre e sinistre.



59.1.6 Esercizio

Con l'ausilio del proprio programma di scrittura preferito, si realizzi una pagina con i margini e le colonne descritti nella figura successiva. Si riempiano le colonne con del testo casuale, allineato simultaneamente a sinistra e a destra, in modo da poter misurare gli spazi ottenuti.

Figura 59.16. Configurazione di una pagina A4, a due colonne, senza fili di separazione.



59.2 Il «paragrafo»

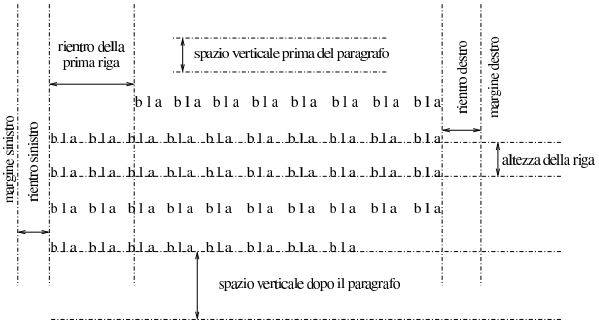
Nei programmi di scrittura, il «paragrafo» è un blocco di testo continuo che si snoda nello spazio orizzontale di cui dispone, andando a capo quando necessario, in modo automatico. Generalmente, il paragrafo del programma di scrittura corrisponde al capoverso, secondo la terminologia tipografica.

Quando si vogliono configurare le caratteristiche di un paragrafo, viene preso in considerazione quello in cui si trova il cursore di scrittura (nel testo), oppure il gruppo di paragrafi evidenziati con il puntatore del mouse.

59.2.1 Dimensioni

Le dimensioni che riguardano il paragrafo sono principalmente: l'altezza tra le righe, i margini verticali ed eventualmente anche orizzontali, rispetto allo spazio a disposizione; inoltre vanno considerati i rientri del testo contenuto, dove quello della prima riga viene trattato separatamente. A questo proposito, va osservato che spesso i margini orizzontali vengono assimilati alla gestione dei rientri.

Figura 59.17. Dimensioni relative alla gestione dei paragrafi.



Una volta selezionato il paragrafo o i paragrafi di proprio interesse, con AbiWord si configurano le dimensioni con *Formato, Paragrafo, Rientri e spaziatura*; con OpenOffice Writer si accede a queste funzioni con *Formato, Paragrafo, Paragrafo*.

Figura 59.18. AbiWord: *Formato, Paragrafo, Rientri e spaziatura.*

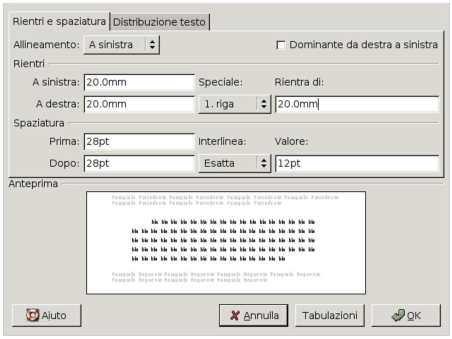
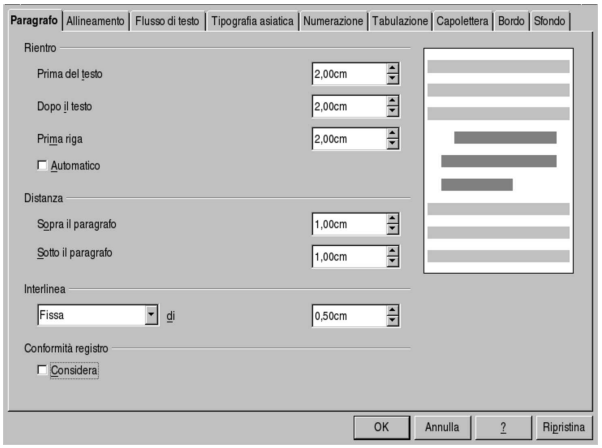


Figura 59.19. OpenOffice Writer: *Formato, Paragrafo, Paragrafo.* In questo caso, le dimensioni sono abbastanza simili allo stesso esempio relativo ad AbiWord.



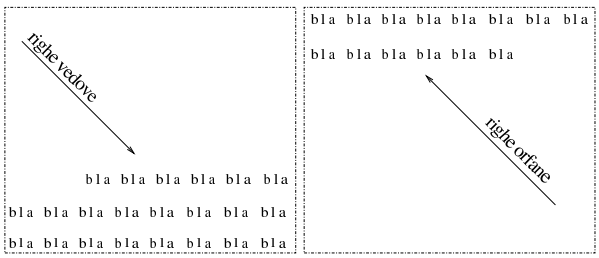
59.2.2 Controllo del flusso del testo

Il testo contenuto in un paragrafo, scorre seguendo lo spazio a sua disposizione. Generalmente, le cose che si possono controllare sono: la sillabazione; il fatto che il paragrafo debba rimanere unito o meno, quando risulterebbe a cavallo di due colonne o di due pagine; la gestione delle righe vedove e di quelle orfane; il fatto che il paragrafo debba rimanere unito al successivo (come nel caso di un titolo che non può apparire da solo alla fine di una pagina).

La sillabazione implica che il testo sia associato a una certa lingua, della quale si conoscono le regole per separare le parole alla fine delle righe. La configurazione della sillabazione può anche richiedere che l'interruzione delle parole non sia troppo frequente o che separi solo nella parte centrale delle parole stesse.

Quando un paragrafo si trova a cavallo tra due colonne o tra due pagine, può risultare spiacevole finire una colonna o una pagina con poche righe, così come è spiacevole iniziare una colonna o una pagina con poche righe di un paragrafo proveniente dalla colonna o da una pagina precedente. Le righe iniziali di un paragrafo che si divide sono le ***vedove***, mentre le righe finali dello stesso paragrafo, sono le ***orfane***.

Figura 59.20. Vedove e orfane.



Una volta selezionato il paragrafo o i paragrafi di proprio interesse, con AbiWord si configura il flusso del testo con *Formato, Paragrafo, Distribuzione testo*; con OpenOffice Writer si accede a queste funzioni con *Formato, Paragrafo, Flusso di testo.*

Figura 59.21. AbiWord: *Formato, Paragrafo, Distribuzione testo.* Si può osservare che la gestione del controllo di vedove e orfane, si ottiene selezionando la casella '**`controllo righe isolate`**'.

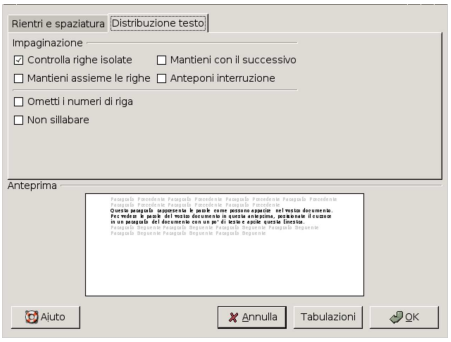
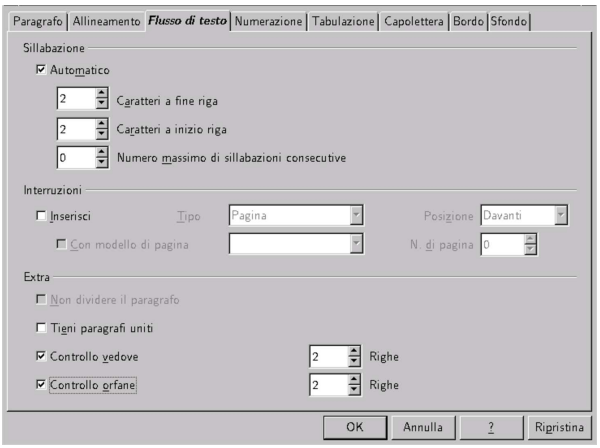


Figura 59.22. LibreOffice Writer: *Formato, Paragrafo, Flusso di testo.*

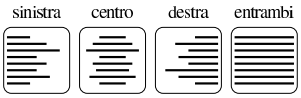


59.2.3 Allineamento

Quando si scrive nel contesto di una lingua occidentale, il testo di un paragrafo si trova normalmente allineato a sinistra. Di norma è consentito allineare il testo in modo diverso: al centro, a destra, o simultaneamente a sinistra e a destra.

L'allineamento del testo è un problema frequente, per cui i programmi di scrittura visuali consentono spesso di accedervi attraverso pulsanti grafici a icona.

Figura 59.23. Icone tipiche per la selezione rapida degli allineamenti comuni.



A parte il caso dell'uso di icone pronte per gestire l'allineamento, con AbiWord si può utilizzare il menù con *Formato, Paragrafo, Rientri e spaziatura*, mentre con OpenOffice Writer le voci di menù da seguire sono *Formato, Paragrafo, Allineamento.*

Figura 59.24. AbiWord: *Formato, Paragrafo, Rientri e spaziatura.*

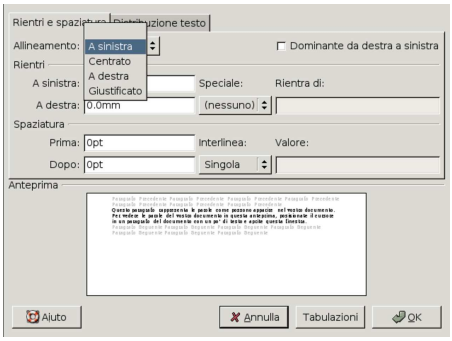
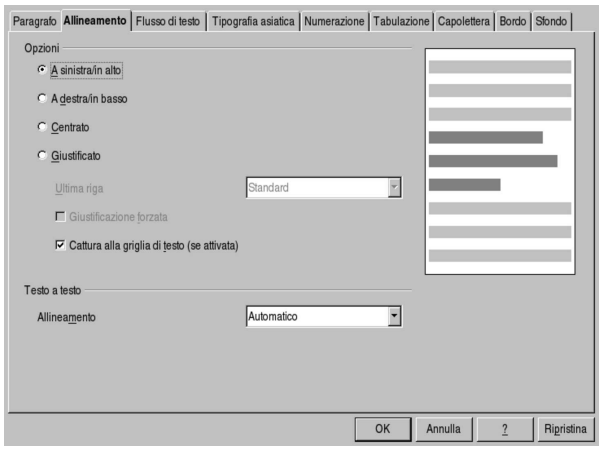


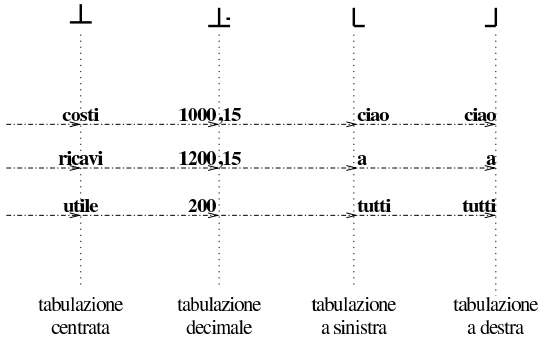
Figura 59.25. LibreOffice Writer: *Formato, Paragrafo, Allineamento.*



59.2.4 Tabulazione

Attraverso gli ***stop di tabulazione***, si possono fissare delle posizioni orizzontali da raggiungere facilmente con l'inserimento di un carattere di tabulazione, il quale si ottiene normalmente con il tasto [*Tab*]. Lo stop di tabulazione che si fissa permette generalmente di stabilire anche il flusso del testo a partire dalla sua posizione; in pratica si può specificare un allineamento sinistro, destro, al centro e decimale. Per indicare gli stop di tabulazione esistenti, i programmi usano una simbologia abbastanza uniforme, dove si intende anche l'allineamento che si ottiene.

Figura 59.26. Tabulazioni e simbologia comune.



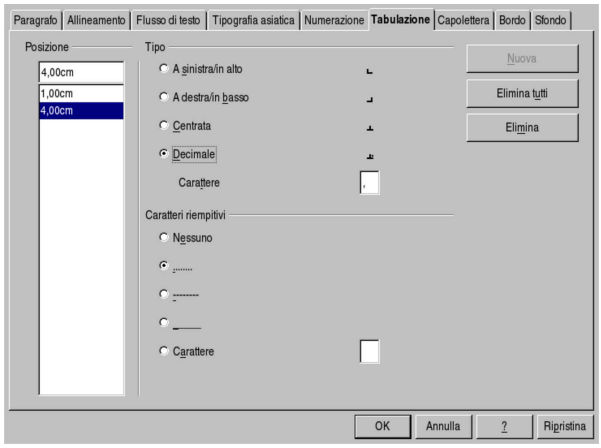
Oltre alla scelta dell'allineamento, in corrispondenza dello spazio che separa il testo dalla tabulazione successiva, è possibile fare apparire un filo o una sequenza di puntini, che si usano di solito per facilitare la lettura.

Nei programmi più semplici e limitati, le tabulazioni potrebbero essere associate al documento nella sua interezza, ma generalmente queste si abbinano ai paragrafi, in modo tale da poter cambiare lo schema di tabulazione di volta in volta. Come per tutte le funzioni che riguardano i paragrafi, prima di intervenire sulla configurazione delle tabulazioni, occorre posizionare il cursore di scrittura o selezionare una porzione di testo.

Con AbiWord si configurano le tabulazioni attraverso *Formato, Tabulazioni*, mentre con LibreOffice Writer si deve selezionare *Formato, Paragrafo, Tabulazione.*

Figura 59.27. AbiWord: *Formato, Tabulazioni.*

Figura 59.28. LibreOffice Writer: *Formato, Paragrafo, Tabulazione.*

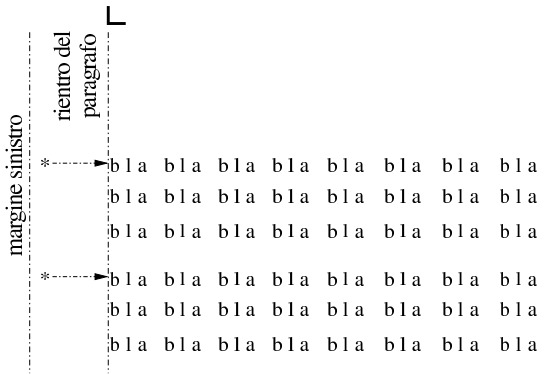


Dalle figure si può osservare che può essere necessario chiarire quale simbolo viene usato per individuare la posizione corretta dell'allineamento decimale: il punto o la virgola.

59.2.5 Elenchi

I programmi di scrittura gestiscono gli elenchi, puntati o numerati, attraverso un meccanismo specifico di gestione dei rientri del testo. In pratica, se un programma di scrittura non disponesse della gestione degli elenchi, questi si potrebbero ottenere aumentando il margine sinistro dei paragrafi, facendo rientrare all'indietro la prima riga, combinando uno stop di tabulazione della stessa lunghezza del margine del paragrafo stesso.

Figura 59.29. Creazione di un elenco puntato con la sola gestione dei rientri e delle tabulazioni, mettendo un asterisco seguito da un carattere di tabulazione nella prima riga.



Di norma, gli elenchi puntati e numerati vengono gestiti automaticamente dai programmi di scrittura, pertanto è il programma che aggiunge il pallino o il numero iniziale. Eventualmente, quando un paragrafo deve continuare allo stesso livello di rientro di una voce dell'elenco, senza avere il puntino o il numero iniziale, basta cancellare manualmente il simbolo in questione, come se fosse un carattere qualsiasi.

Con AbiWord e con LibreOffice Writer si crea un elenco puntato o numerato attraverso la selezione di *Formato, Elenchi puntati e numerati.*

Figura 59.30. AbiWord: *Formato, Elenchi puntati e numerati.*

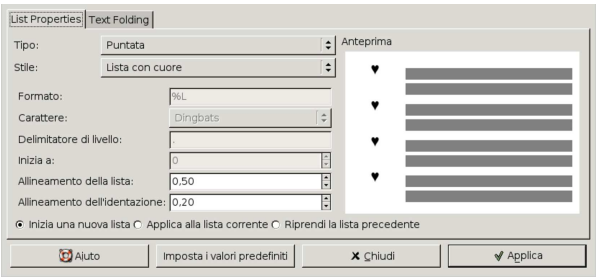
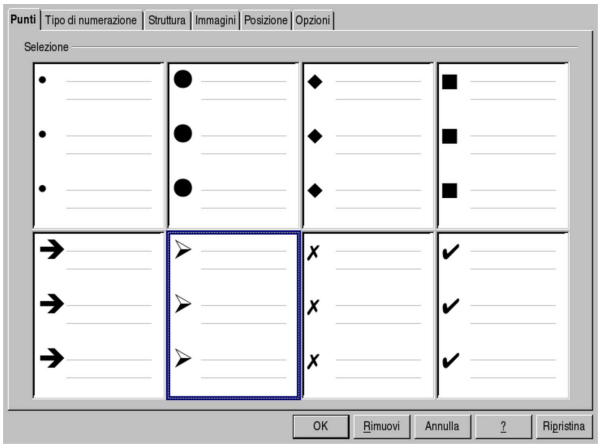


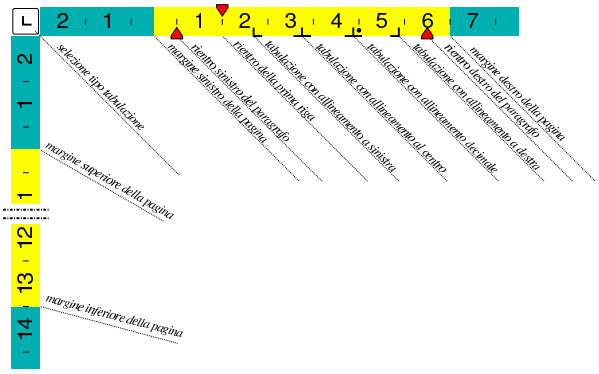
Figura 59.31. LibreOffice Writer: *Formato, Elenchi puntati e numerati.*



59.2.6 Righello

Generalmente i programmi di scrittura visuale collocano un righello sulla parte superiore e sulla parte sinistra, in modo da facilitare l'individuazione e la modifica di margini, rientri e tabulazioni. Va osservato che di norma, con il righello non è possibile intervenire sulle spaziature verticali relative al testo e ai paragrafi.

Figura 59.32. Righello.



59.3 Il testo

Di norma, i programmi di scrittura visuali consentono di intervenire nel testo con molta libertà. Generalmente si tratta di modifiche che riguardano il tipo di carattere, con tutte le sue caratteristiche, compreso il corpo.

Quando si vogliono configurare le caratteristiche di una porzione di testo, è necessario prima evidenziarlo con il puntatore del mouse, oppure attraverso tecniche simili con l'ausilio della tastiera.

59.3.1 La specie alfabetica e il tipo di carattere

Nella tradizione tipografica, il carattere da stampa viene classificato, prima di tutto, in base alla specie alfabetica, ovvero in base al tipo di scrittura a cui è destinato: latino, cirillico, greco, arabo,... Nella tipografia basata sull'uso di software, secondo le convenzioni degli anni 1990, mancando una codifica univoca per ogni tipo di simbolo tipografico, era possibile sostituire la specie alfabetica, ottenendo caratteri alternativi. Pertanto, era possibile scrivere un testo con caratteri latini e poi trasformarlo in simboli, attribuendo successivamente la specie simbolica.

Nella tipografia attuale, ogni simbolo di ogni specie alfabetica ha un proprio codice distinto, secondo quella che è nota come codifica universale, pertanto, i programmi di scrittura non dovrebbero più consentire la sostituzione della specie alfabetica, perché ciò sarebbe un'operazione priva di significato.

Quello che si può cambiare è il tipo di carattere, inteso come gruppo stilistico (Times, Courier, Helvetica, ecc.). Tuttavia, difficilmente un tipo di carattere contiene tutti i simboli annoverati dalla codifica universale; pertanto, il cambiamento del tipo di carattere (del gruppo stilistico) può comportare la possibilità o meno di scrivere utilizzando certi simboli.

Per comprendere il problema legato alla scelta del tipo di carattere, in relazione alla specie alfabetica, si supponga di voler scrivere utilizzando l'alfabeto cirillico, naturalmente disponendo di una tastiera configurata correttamente per il cirillico. Senza pensare al problema della specie alfabetica, si potrebbe scegliere un carattere qualunque, che però contiene soltanto i simboli utili per l'alfabeto latino. In questo caso, il programma di scrittura potrebbe limitarsi a non visualizzare i caratteri della scrittura cirillica.

59.3.2 Gruppo stilistico, serie e corpo

Le cose più comuni che si definiscono del carattere sono: il gruppo stilistico, ovvero il nome del carattere; la serie, ovvero la forma; il corpo, ovvero la dimensione. Purtroppo, la terminologia utilizzata dai programmi di scrittura visuale non rispetta la tradizione tipografica e questo si ripercuote in un utilizzo errato della lingua italiana.

Con AbiWord e con LibreOffice Writer si interviene nelle caratteristiche principali del carattere con la selezione di *Formato, Carattere, Carattere.*

Figura 59.33. AbiWord: *Formato, Carattere, Carattere*. Si vede la selezione di un carattere Free Times, tondo, alto 24 punti.

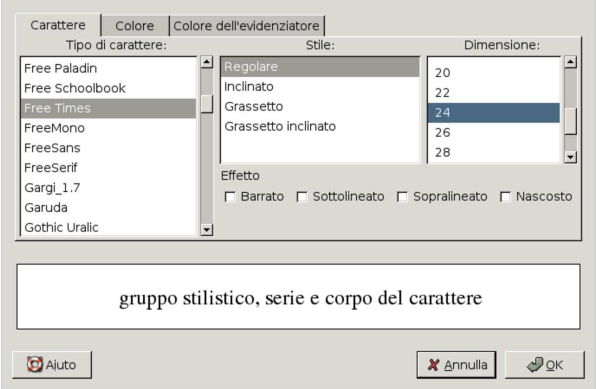
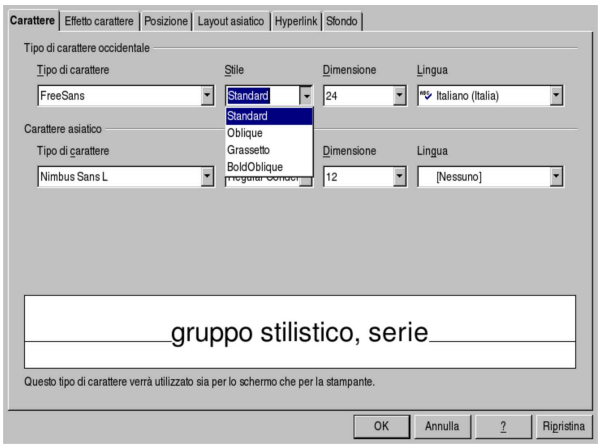


Figura 59.34. LibreOffice Writer: *Formato, Carattere, Carattere*. Si vede la selezione di un carattere FreeSans, tondo, alto 24 punti.



59.3.3 Effetti comuni applicabili al carattere

Quando si pensa al carattere tipografico, si tende normalmente a concepire un testo nero su un foglio bianco. Naturalmente, le possibilità di un sistema tipografico basato su software sono molte di più. Gli effetti più comuni che si possono applicare al testo consistono nel cambiamento del colore, nello spostamento ad apice o a pedice, nell'aggiunta di sottolineature, sopralineature o barrature. Evidentemente, l'applicazione di effetti speciali al testo sconfina poi con la grafica.

Generalmente, con i programmi di scrittura visuale, gli effetti associati al carattere si ottengono a partire dalla voce di menù *Formato*, *Carattere*, ma nel dettaglio ognuno ha poi un suo modo di classificarli.

Figura 59.35. AbiWord: *Formato, Carattere, Carattere*. Si vede la selezione di un carattere FreeMono, tondo, alto 24 punti, sottolineato, barrato e sopralineato.

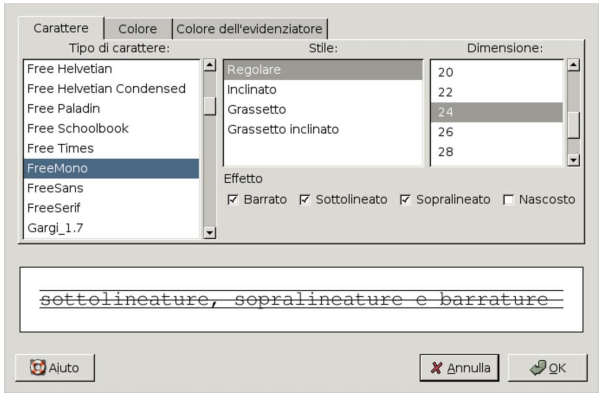


Figura 59.36. AbiWord: *Formato, Carattere, Colore*.

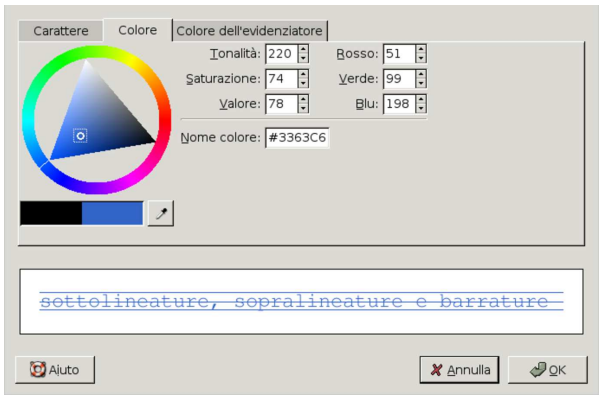


Figura 59.37. AbiWord: *Formato, Formattazione del testo*. Da questo sottomenù si possono controllare alcuni effetti del carattere, in particolare lo spostamento ad apice o a pedice.

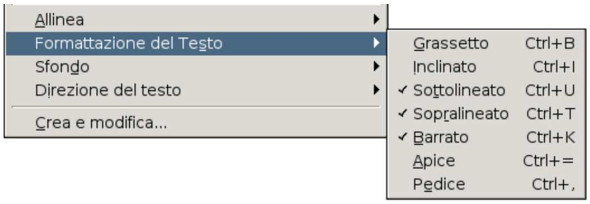


Figura 59.38. LibreOffice Writer: *Formato, Carattere, Effetto carattere*. Si vede la selezione di un carattere di colore normale (nero), con una sottolineatura tratteggiata di colore blu, di una barratura normale, limitata alle parole e di una specie di sopralineatura con l'aggiunta di cerchietti.

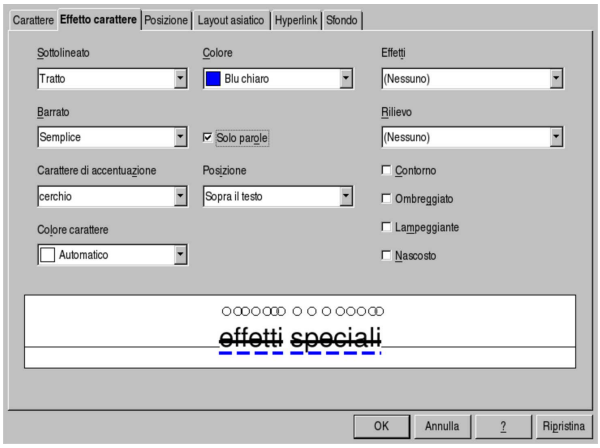
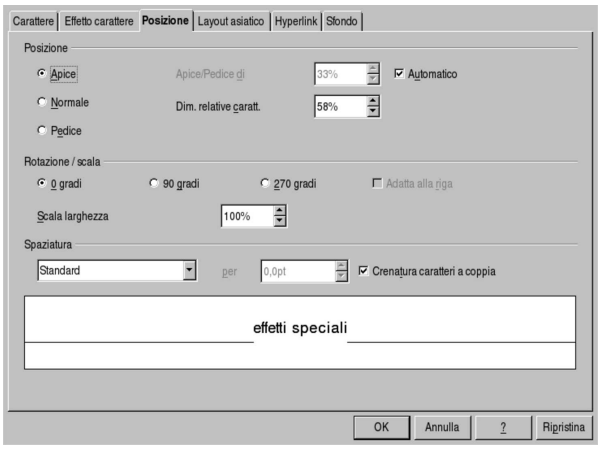


Figura 59.39. LibreOffice Writer: *Formato, Carattere, Posizione*. Si vede la selezione di uno spostamento ad apice. Nella finestra si può notare anche la possibilità di ruotare il testo.



59.3.4 Maiuscole e minuscole

Nello scrivere si può incontrare la necessità di modificare sistematicamente il testo, per ciò che riguarda la scelta di lettere maiuscole o minuscole. Questo cambiamento può dover essere permanente, nel senso che si vuole che il programma sostituisca le lettere, dove necessario, oppure può essere applicato come se fosse un effetto da associare al testo.

Figura 59.40. AbiWord: *Formato, Cambia MAIUSCOLE/minuscole*. In questo caso, si tratta di eseguire una sostituzione permanente del testo.

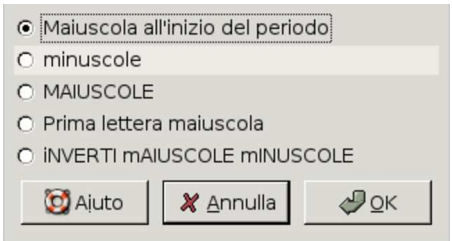
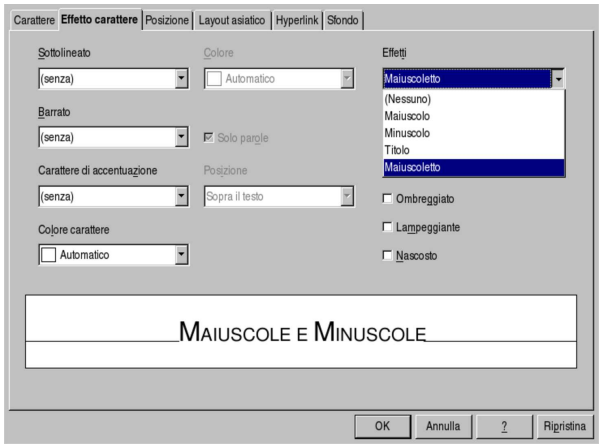


Figura 59.41. LibreOffice Writer: menù *Formato, Maiuscole/minuscole*, per la sostituzione permanente del testo.

Figura 59.42. LibreOffice Writer: *Formato, Carattere, Effetto carattere*. Da questa scheda è possibile controllare anche l'associazione di effetti legati all'uso di maiuscole e minuscole, come il maiuscoletto che si vede selezionato.



59.3.5 Direzione del testo

Il programma di scrittura visuale dovrebbe mostrare il testo cambiandone la direzione in base alla specie alfabetica. In pratica, l'autore dovrebbe digitare i simboli secondo la sequenza normale di scrittura per quella data lingua, mentre il programma dovrebbe disporli secondo la direzione appropriata. Per fare un esempio più semplice, quando di scrive utilizzando la specie alfabetica latina, greca o cirillica, il testo deve scorrere da sinistra a destra; quando invece si scrive usando l'alfabeto ebraico o arabo, il testo deve scorrere da destra verso sinistra.

Dal momento che normalmente si usa la codifica universale, la scelta della specie alfabetica (ovvero dell'alfabeto) è una questione che riguarda semplicemente la configurazione della tastiera. Al massimo, può succedere che un certo gruppo stilistico non contenga i simboli necessari per l'alfabeto di proprio interesse, rendendo così necessaria la scelta di un gruppo stilistico adatto.

Figura 59.43. Quattro tipi di specie alfabetiche nello stesso flusso di testo. Sono evidenziate le direzioni del testo dei vari tipi di scrittura.

59.3.6 Controllo ortografico

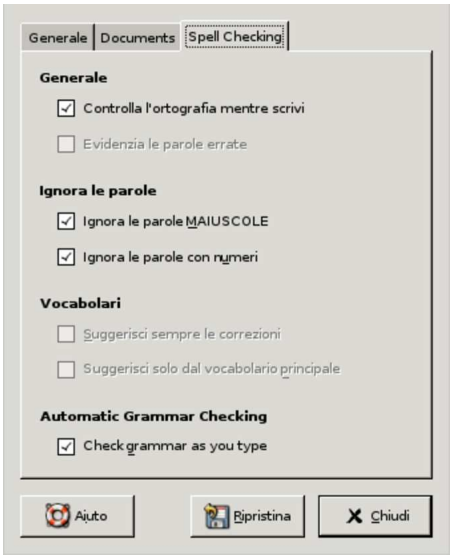
Una delle funzioni più comuni di un programma di scrittura visuale è il controllo ortografico, basato sulla corrispondenza delle parole con un dizionario, inteso come elenco preesistente di parole ritenu-

te valide. In pratica, si tratta di un controllo di pura esistenza delle parole, privo di qualunque verifica sintattica.

Perché il controllo ortografico possa essere di una qualche utilità, è necessario distinguere il dizionario di riferimento in base alla lingua del documento. Di solito, la scelta del linguaggio predefinito dei documenti proviene dalla configurazione locale del sistema operativo (nei sistemi Unix si tratta della variabile di ambiente ***LANG*** ed eventualmente di altre variabili ***LC_****) e può essere modificato secondo la configurazione del programma stesso. Alcuni programmi consentono di specificare un linguaggio unico, altri permettono di attribuire il linguaggio a porzioni di testo, in modo differenziato.

Per quanto riguarda AbiWord, la configurazione relativa al controllo ortografico si raggiunge con *Modifica, Opzioni, Spell Checking*, come si vede nella figura successiva.

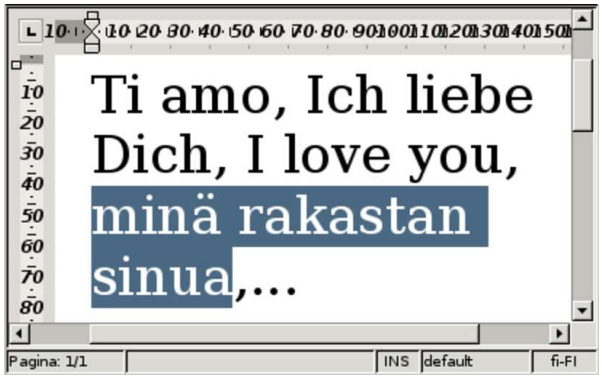
Figura 59.44. AbiWord: *Modifica, Opzioni, Spell Checking.*



Come si può osservare dalla figura, da questa finestra non si accede alla definizione della lingua da usare, perché AbiWord consente di indicare la lingua attraverso un'altra funzione, dove la si può dichiarare per tutto il documento, oppure per la sola porzione di testo selezionata.

Figura 59.45. AbiWord: *Strumenti, Lingua.* In questo modo è possibile definire la lingua predefinita del documento, oppure soltanto quella del testo selezionato.

Figura 59.46. AbiWord durante il funzionamento: si può vedere in basso a destra l'indicazione della lingua di riferimento per il testo selezionato. In questo caso la sigla è '**`fi-FI`**', pertanto la lingua è il finlandese.



LibreOffice Writer ha un sistema di controllo ortografico molto sofisticato, con la possibilità di attivare una correzione automatica in fase di scrittura; tuttavia manca la possibilità di attribuire il linguaggio particolare a delle porzioni di testo. Si accede alla configurazione della lingua da usare ai fini del controllo o della correzione ortografica, attraverso *Strumenti, Opzioni, Impostazioni lingua.* Da qui, la prima cosa da fare è dichiarare la lingua per il controllo ortografico, selezionando la voce *Lingue*; successivamente, si può scegliere invece la voce *Linguistica*, per la configurazione della modalità di intervento del controllo o della correzione ortografica.

Figura 59.47. LibreOffice Writer: *Strumenti, Opzioni, Impostazioni lingua, Lingue.* Per le funzioni del controllo ortografico, conta solo la parte destra centrale, nel riquadro con il titolo «Lingue standard per i documenti». Si può osservare che se la lingua scelta contiene la sigla «ABC», è possibile ottenere il controllo e la correzione ortografica.

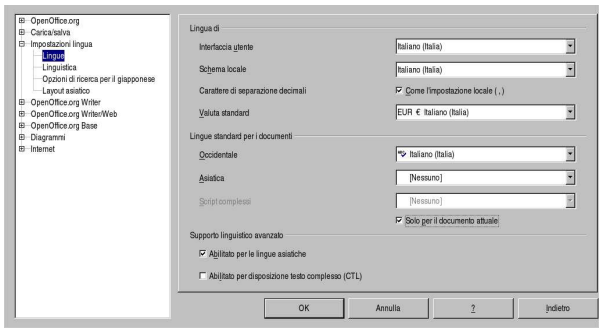
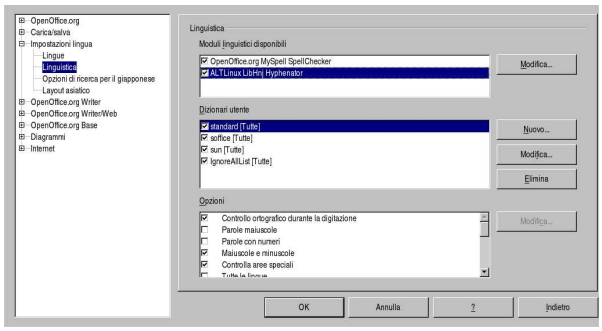


Figura 59.48. LibreOffice Writer: *Strumenti, Opzioni, Impostazioni lingua, Linguistica.*



59.3.7 Correzione ortografica

«

In generale, è spiacevole che il programma di scrittura si permetta di correggere automaticamente quelli che «lui» ritiene essere errori di ortografia. Per esempio, diventa veramente scomodo quando un programma del genere mette a posto le maiuscole all'inizio dei periodi, perché non è sempre vero che ciò vada fatto nel modo in cui «lui» considera. Tuttavia, ci sono certe forme di correzione che diventano utili, in quanto richiamate espressamente e sottoposte al controllo dell'utilizzatore.

Di solito, il programma di scrittura viene configurato in modo da segnalare con una sottolineatura ondulata le parole che non rientrano nel vocabolario previsto. Così, l'utilizzatore può scegliere se correggerle o se lasciarle stare. Quando però si scrive con una lingua che non si conosce bene, può diventare conveniente chiedere un piccolo aiuto al programma, per avere il suggerimento delle parole alternative che potrebbero essere più corrette.

Di norma, quando una parola viene segnalata come errata o sconosciuta, è possibile portarvi sopra il puntatore del mouse e premere il tasto destro, per ottenere una tendina con i suggerimenti per la correzione. Oltre al problema di una lingua che non si conosce bene, in questo modo è possibile sopperire anche alla carenza di simboli della propria tastiera, che potrebbe essere inadeguata per quella lingua.

Figura 59.49. AbiWord mentre suggerisce la sostituzione di una parola finlandese.

Figura 59.50. LibreOffice Writer mentre suggerisce la sostituzione di una parola tedesca.

59.3.8 Simboli speciali e il limite della tastiera

«

Durante la scrittura è facile incontrare la necessità di inserire caratteri che la propria tastiera non consente di scrivere. Di solito i programmi di scrittura offrono la possibilità di inserire un carattere, cercandolo all'interno di un insieme limitato di simboli.

Figura 59.51. AbiWord: *Inserisci, Simbolo*. Il programma consente si scegliere l'insieme di caratteri e offre la possibilità di selezionare uno dei primi simboli di tale insieme.

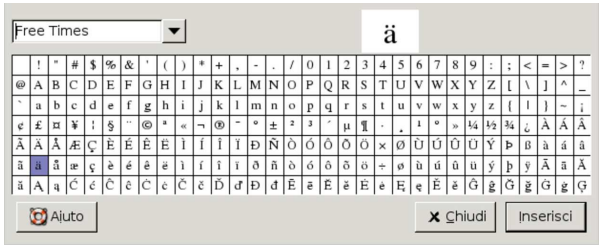
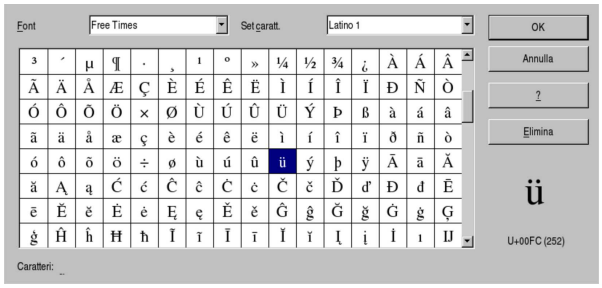
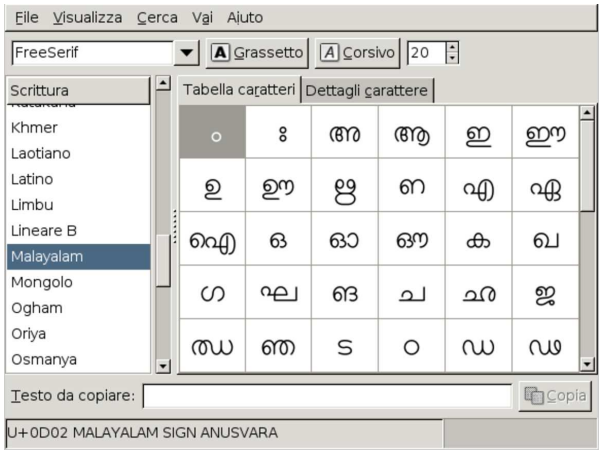


Figura 59.52. LibreOffice Writer: *Inserisci, Carattere speciale*. Il programma consente si scegliere l'insieme di caratteri e offre la possibilità di selezionare uno, all'interno di un sottoinsieme di uso più comune.



Purtroppo, questo inserimento guidato dai programmi di scrittura, di solito limita l'accesso a un certo gruppo di simboli comuni. Per ovviare all'inconveniente, occorre servirsi di un programma specifico, come per esempio Gucharmap.[1]

Figura 59.53. Gucharmap.



A ogni modo, prima di rivolgersi a funzioni o a programmi del genere, è bene sfruttare completamente le possibilità della propria tastiera. Quando si utilizza il sistema grafico X, la configurazione italiana della tastiera prevede diversi accenti morti, che consentono di scrivere con la maggior parte delle lingue europee. La tabella successiva elenca le combinazioni e le sequenze di tasti da premere per ottenere i simboli più comuni.

Tabella 59.54. Funzioni speciali di una tastiera italiana con X.

Tasti	Descrizione o risultato
[*fissamaiuscole*]	Attivando le maiuscole con il tasto [*Fissamaiuscole*], si ottiene la conversione in maiuscolo di tutti i simboli che possono essere rappresentati in questo modo. Per esempio, la lettera «É» si ottiene con le maiuscole inserite e selezionando la lettera «é», cosa che richiede anche l'uso del tasto delle maiuscole normali.
[*AltGr e*], [*AltGr E*]	€, ¢
[*AltGr R*]	®
[*AltGr o*], [*AltGr O*]	ø, Ø
[*AltGr a*], [*AltGr A*]	æ, Æ
[*AltGr s*], [*AltGr S*]	ß, §
[*AltGr <*], [*AltGr >*]	«, »
[*AltGr c*], [*AltGr C*]	¢, ©
[*AltGr n*], [*AltGr N*]	ñ, Ñ
[*AltGr ,*]	Accento acuto morto
[*AltGr ,*][*a*], [*AltGr ,*][*A*], [*AltGr ,*][*i*], [*AltGr ,*][*I*],...	á, Á, í, Í,...
[*AltGr :*]	Dieresi morta
[*AltGr :*][*a*], [*AltGr :*][*A*], [*AltGr :*][*e*], [*AltGr :*][*E*],...	ä, Ä, ë, Ë,...
[*AltGr ^*]	Accento circonflesso morto
[*AltGr ^*][*a*], [*AltGr ^*][*A*], [*AltGr ^*][*e*], [*AltGr ^*][*E*],...	â, Â, ê, Ê,...
[*AltGr §*], [*AltGr maiuscole ù*]	Accento *breve* morto
[*AltGr §*][*a*], [*AltGr §*][*A*], [*AltGr §*][*u*], [*AltGr §*][*U*],...	ă, Ă, ŭ, Ŭ,...
[*AltGr °*], [*AltGr maiuscole à*]	Accento tondo morto
[*AltGr °*][*a*], [*AltGr °*][*A*], [*AltGr °*][*u*], [*AltGr °*][*U*],...	å, Å, ů, Ů,...
[*AltGr -*]	Accento *macron* morto
[*AltGr -*][*a*], [*AltGr -*][*A*], [*AltGr -*][*e*], [*AltGr -*][*E*],...	ā, Ā, ē, Ē,...

Figura 59.55. Mappa della tastiera italiana secondo X, con gli accenti morti in evidenza.

59.3.9 Visualizzazione dei «caratteri non stampabili»

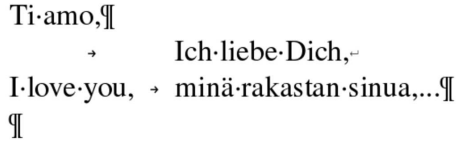
Quando si scrive un testo, lo spazio occupato dai caratteri viene creato contestualmente con la scrittura. In altri termini, dove non ci sono dei caratteri, anche se invisibili, c'è il vuoto. Oltre ai caratteri tipografici veri e propri, durante la scrittura si inseriscono dei codici che hanno una loro funzionalità, ma non devono essere resi tipograficamente nella stampa. Per esempio, il codice che identifica l'inserimento di una tabulazione, serve ad allineare il testo, ma non si rappresenta; nello stesso modo, lo spazio (normale o non interrompibile) esiste, ma non deve essere reso tipograficamente.

Durante la scrittura visuale, può essere utile avere un modo per individuare la presenza dei codici che in fase di stampa non devono essere resi, per sapere cosa è stato inserito realmente e per sapere dove arriva lo spazio usato dal testo.

Figura 59.56. Icona tipica per attivare o disattivare la visualizzazione dei simboli non stampabili.

Di norma, la funzione con cui si abilita o si disabilita la visualizzazione di questi simboli si raggiunge con *Visualizza*, *Caratteri non stampabili*, oppure con *Visualizza*, *Paragrafi*, o con altre varianti simili.

Figura 59.57. Un testo con i caratteri non stampabili in evidenza.



59.3.10 Inserzione di codici con funzioni speciali

Come accennato nella sezione precedente, durante la scrittura si inseriscono anche codici che non corrispondono a dei caratteri da stampa veri e propri, in quanto rappresentano una funzione. Nelle situazioni più comuni si tratta dello spazio, delle tabulazioni e della conclusione dei paragrafi, ai quali si aggiungono poi quei codici che servono a interrompere il testo in qualche modo: per mandarlo a capo, pur non avendo concluso il paragrafo; per riprendere in una colonna successiva o in una pagina successiva.

I codici delle funzioni più comuni si ottengono generalmente con la tastiera, mentre quelli di «interruzione», spesso sono accessibili solo tramite il menù; di solito a partire da *inserisci*, *interruzione*.

Tabella 59.58. Inserimento di «caratteri non stampabili» attraverso la tastiera.

Tasti	Descrizione o risultato
[*barra-spaziatrice*]	Spazio.
[*Tab*]	Tabulazione.
[*Invio*]	Conclusione di un paragrafo.
[*Maiuscole Invio*]	Conclusione di una riga (manda a capo il testo, senza concludere il paragrafo).
[*Ctrl Invio*]	Conclusione della pagina (conclude il paragrafo e salta alla pagina successiva).

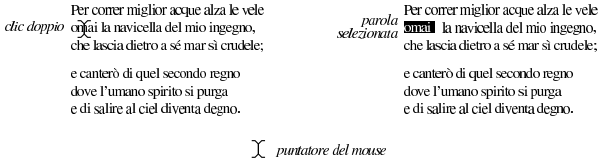
59.4 Selezione, spostamento, copia

Spesso, alcune operazioni comuni, come quelle che servono a tagliare, copiare e incollare il testo, possono essere svolte in vari modi alternativi, ma di norma l'utente ne conosce uno solo; in particolare l'utente tipico ignora completamente le possibilità della tastiera.

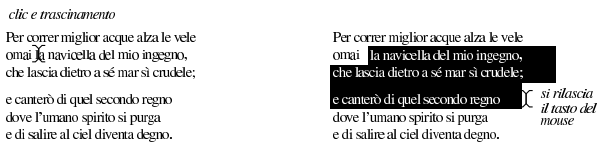
59.4.1 Selezione

La selezione di una porzione del documento serve per potervi poi associare un'operazione da compiere. In altri termini, con la selezione si individua l'ambito a cui successivamente vanno applicate delle funzioni.

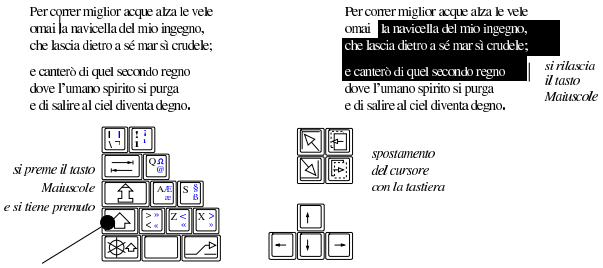
La selezione di una sola parola può essere ottenuta con un clic doppio del primo tasto del mouse, quando il puntatore si trova sulla parola stessa:



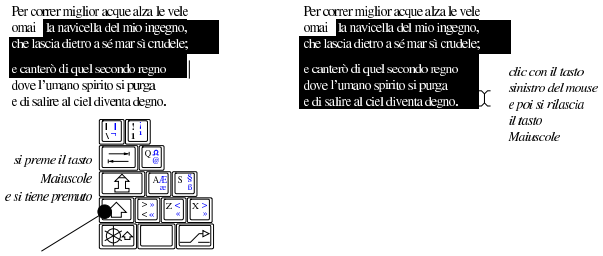
La selezione di una porzione continua del documento si può ottenere con l'uso del mouse, premendo il tasto sinistro quando il puntatore si trova all'inizio della zona da selezionare, trascinando fino all'inizio della zona (per trascinamento si intende lo spostamento del puntatore del mouse, mentre si continua a tenere premuto il tasto sinistro). Al termine si rilascia il tasto sinistro. Se l'operazione di selezione viene ripetuta o se si sposta il cursore all'interno del testo, la selezione precedente viene perduta.

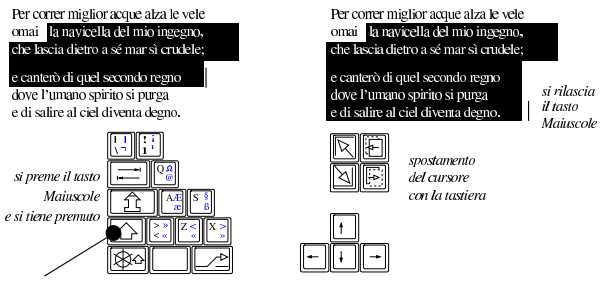


La stessa cosa può essere ottenuta con l'uso della tastiera, portando il cursore all'inizio dell'area da selezionare, premendo quindi il tasto [*Maiuscole*] e, mantenendolo premuto, spostando il cursore fino alla fine dell'area da selezionare:



Una selezione preesistente può essere estesa o ridotta, premendo nuovamente il tasto [*Maiuscole*] e raggiungendo la fine dell'area di proprio interesse. Per raggiungere la posizione finale si può usare la tastiera o il mouse.

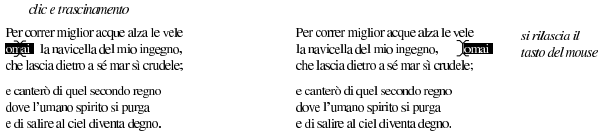




Alcuni programmi consentono anche la selezione di aree multiple, mantenendo premuto il tasto [*Ctrl*]. In tal caso, però, le selezioni devono avvenire con l'ausilio del mouse.

59.4.2 Spostamento

Un'area già selezionata (selezione singola) può essere spostata con l'ausilio del mouse: si porta il puntatore grafico sopra l'area evidenziata, si preme il tasto sinistro e si trascina. Quando si rilascia il tasto sinistro del mouse, l'area selezionata viene collocata a partire dalla posizione in cui si trova il puntatore grafico.



59.4.3 Taglia-copia-incolla

Una volta selezionata un'area di un documento, le operazioni più semplici che si possono compiere consistono nel ritaglio o nella copia di questa per poterla poi incollare in un'altra posizione. Il ritaglio o la copia comportano la memorizzazione in un'area transitoria, dalla quale si possono ottenere quante copie si vogliono.

Per queste operazioni si usano generalmente le funzioni del mouse, ma la tastiera può semplificarle di molto.

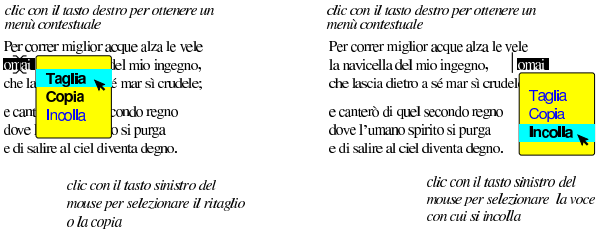
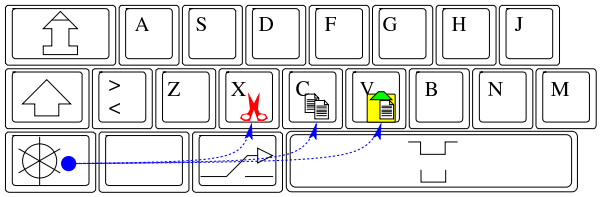


Tabella 59.66. Taglia-copia-incolla, con vari modi alternativi.

Taglia	Copia	Incolla
[*Ctrl x*]	[*Ctrl c*]	[*Ctrl v*]
	[*Ctrl Ins*]	[*Maiuscole Ins*]
		

Figura 59.67. Associazione tra i comandi da tastiera e le icone per tagliare, copiare e incollare.



59.5 Tabelle

I programmi di scrittura visuali consentono normalmente di inserire delle tabelle, che complessivamente prendono il posto che potrebbe occupare un paragrafo, dove all'interno di ogni cella, a loro volta, possono contenere più paragrafi.

In generale, le tabelle sono molto importanti per rappresentare sinteticamente dei dati; tuttavia, spesso la creazione e la modifica di questi componenti è poco pratica, soprattutto perché viene consentito di realizzare dei reticoli disomogenei. Pertanto, le tabelle vanno usate con criterio, per evitare di incontrare difficoltà eccessive rispetto all'obiettivo che ci si prefigge.

59.5.1 Creazione di una tabella

Per creare una tabella, si posiziona prima il cursore del testo nel punto in cui questa deve essere inserita, tenendo conto che ciò comporta l'aggiunta della conclusione di un paragrafo. Pertanto, se si cerca di inserire una tabella a metà di una riga che contiene del testo, quella riga viene divisa.

Generalmente, i programmi di scrittura dispongono del menù *Tabella* o *Tabelle*, per raccogliere le funzioni di creazione e di modifica di questi componenti, ma spesso si aggiunge un'icona per la creazione rapida, nella barra degli strumenti.

Figura 59.68. AbiWord: *Tabelle, Inserisci tabella, Tabella.*

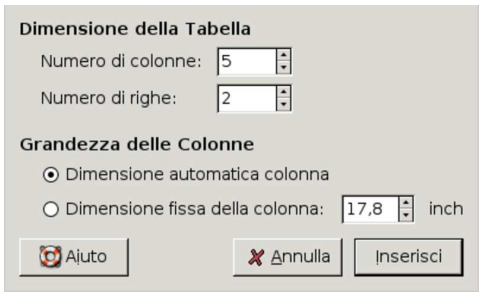
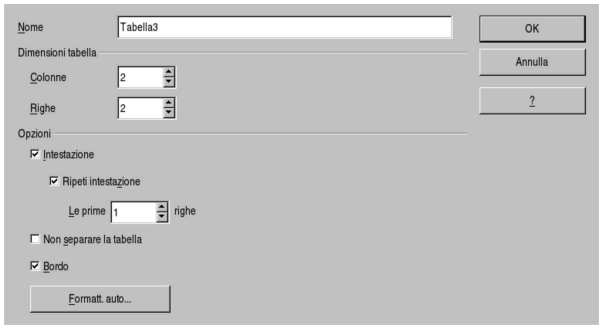


Figura 59.69. LibreOffice Writer: *Tabella, Inserisci, Tabella.*



Generalmente, la tabella che si inserisce va occupare tutto lo spazio che potrebbe prendere un paragrafo di testo normale, pertanto si allinea ai bordi orizzontali, inoltre le colonne assumono una larghezza uguale tra di loro.

Di solito, dopo la creazione, si procede alla modifica della larghezza delle colonne, con l'aiuto del mouse, portando il puntatore a cavallo delle colonne e trascinando. Purtroppo, questo tipo di operazione non comporta dei risultati uniformi tra un programma e l'altro, così può capitare che venga adattata la larghezza della colonna di sinistra, compensando con quella della colonna a destra, oppure spostando le altre colonne di conseguenza. Così facendo, l'inizio della prima colonna può essere portato anche oltre il margine sinistro e la fine dell'ultima colonna può andare oltre il margine destro. Ovviamente questi debordamenti sono da evitare se si vuole ottenere un risultato estetico gradevole.

59.5.2 Interazione con le tabelle

In presenza di una tabella, le funzioni della tastiera e del mouse cambiano leggermente, in modo abbastanza intuitivo. Per quanto riguarda la tastiera, il tasto che di norma serve a inserire delle tabulazioni in un testo normale [*Tab*], nella tabella serve a passare alla cella successiva, oppure a quella precedente se si aggiungono le maiuscole [*Maiuscole Tab*]. Inoltre, quando ci si trova nell'ultima cella della tabella, l'uso del tasto [*Tab*] comporta la creazione implicita di una nuova riga.

Quando si vuole selezionare il contenuto di una riga o di una colonna di una tabella, si avvicina il puntatore del mouse all'inizio della riga (di solito a sinistra) o all'inizio della colonna (di solito in alto) e quindi si seleziona. Una volta selezionata una riga o una colonna, spesso, con il tasto destro del mouse si ottiene un menù contestuale.

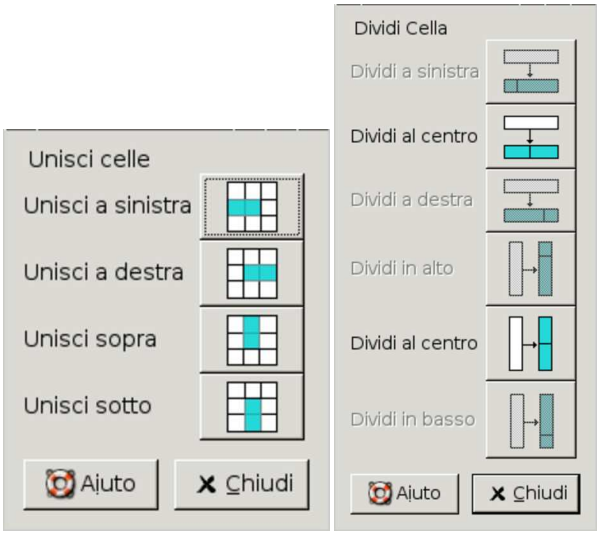
59.5.3 Righe, colonne e fusione delle celle

Una volta creata una tabella, è normale trovarsi a doverne modificare la struttura di righe e colonne, così come può capitare l'utilità di unire assieme più celle.

Per gli interventi su righe o colonne, generalmente si seleziona prima una riga o una colonna, per dare poi l'ordine di eliminazione o di inserimento. Queste funzioni sono accessibili normalmente dal menù *Tabella* o *Tabelle*, oppure, spesso anche dal menù contestuale che si ottiene di solito premendo il tasto destro del mouse.

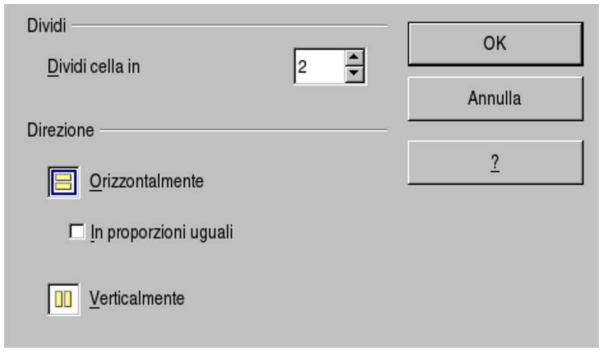
Per la fusione delle celle, il procedimento dipende molto dal programma. Con LibreOffice Writer è necessario selezionare le celle da unire, usando il mouse come se si volesse selezionare del testo, quindi si deve accedere al menù: *Tabella, Collega celle*. Con AbiWord occorre mettere il cursore del testo nella cella da espandere, senza selezionare alcunché, quindi, dal menù si deve selezionare *Tabelle, Unisci celle*, dove si deve indicare da che parte si vuole espandere la cella attuale.

Figura 59.70. AbiWord: a sinistra *Tabelle, Unisci celle*; a destra *Tabelle, Separa celle*. I pulsanti grafici che si vedono nella finestra, possono essere usati ripetutamente, per unire o dividere le celle, a passi successivi.



Contrariamente a ciò che si potrebbe pensare, di norma, la separazione delle celle è una funzione che riguarda qualunque cella, anche quelle che non hanno subito in precedenza alcuna fusione. Quindi, attraverso le funzioni di separazione, è possibile costruire tabelle con griglie molto disomogenee. Naturalmente, per questo tipo di operazione, occorre avere prima selezionato la cella da dividere, cosa che si ottiene normalmente collocandoci dentro il cursore del testo.

Figura 59.71. LibreOffice: *Tabella, Dividi celle*. Nella finestra che si ottiene, occorre specificare da che parte dividere lo spazio della cella.



59.5.4 Bordi e sfondi

Le caratteristiche più comuni che si possono regolare nelle celle di una tabella sono i bordi e gli sfondi. Ciò permette di evidenziare i contenuti in modo da facilitarne l'interpretazione.

Figura 59.72. AbiWord: *Tabelle, Formato tabella*.

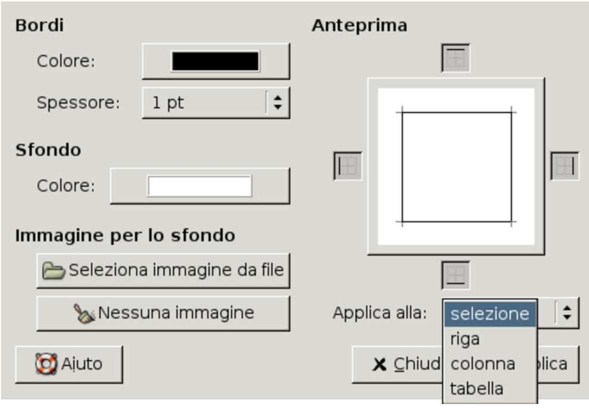


Figura 59.73. LibreOffice: *Tabella, Proprietà tabella, Bordo*.

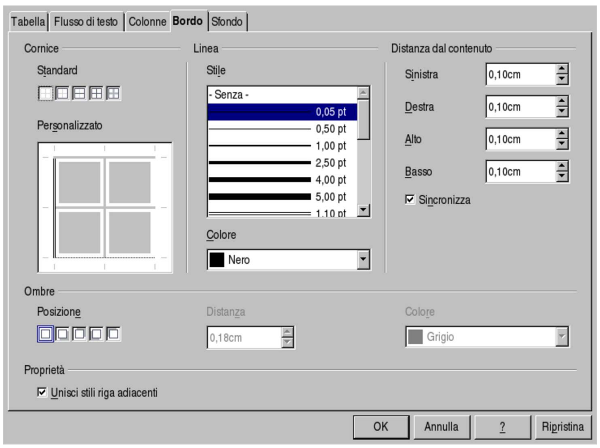
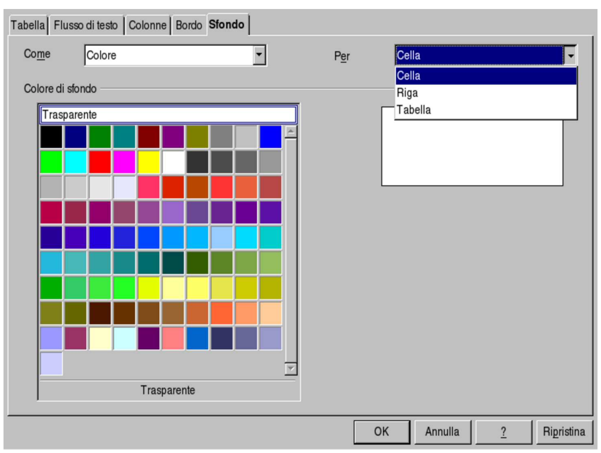


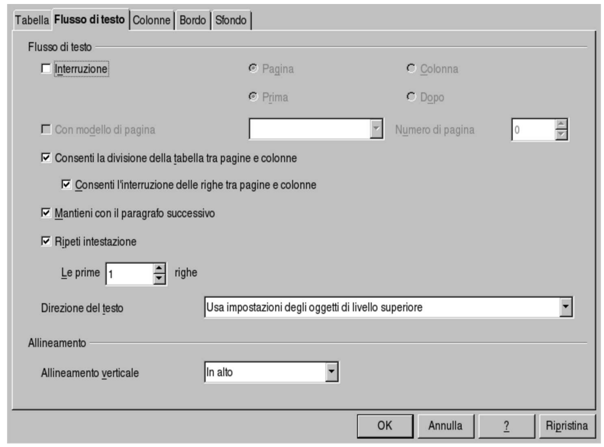
Figura 59.74. LibreOffice: *Tabella, Proprietà tabella, Sfondo*.



59.5.5 Righe di intestazione

Quando si realizza una tabella e si vuole consentire che questa si divida tra le pagine o tra le colonne della pagina, di solito è utile fare in modo che una o più righe iniziali vengano ripetute dopo il salto pagina o dopo il cambiamento di colonna. In pratica, conviene fare in modo che vengano ripetute le intestazioni delle colonne della tabella. Si tratta comunque di una funzionalità che non è sempre presente, ma se c'è è bene sfruttarla.

Figura 59.75. LibreOffice: *Tabella, Proprietà tabella, Flusso di testo*.



59.5.6 Copia dal foglio elettronico

Quando si copia un lavoro fatto con un foglio elettronico nel programma di scrittura, il risultato che si ottiene può essere di tipo differente, a seconda della compatibilità reciproca che c'è tra i due programmi.

Nella migliore delle ipotesi si ottiene l'inserzione di un «oggetto», che si può gestire come un'immagine, ma che funziona effettivamente come nel foglio elettronico di origine. In questo caso dovrebbero persistere le funzionalità originali, soprattutto le formule.

Quando la tabella del foglio elettronico non si può incorporare nel testo, si potrebbe ottenere una trasformazione in una tabella normale per un programma di scrittura, oppure, si potrebbe arrivare a una degradazione in forma di immagine pura e semplice.

59.6 Incorporazione di oggetti

Ciò che si inserisce normalmente in un documento realizzato con un programma di scrittura visuale è del testo raggruppato in paragrafi, che eventualmente può essere organizzato in forma tabellare. Per aggiungere al documento qualche componente in più, come per esempio nel caso di una figura, occorre definire uno spazio rettangolare, che in qualche modo va ancorato nel documento.

I componenti rettangolari che si inseriscono in un documento possono essere considerati genericamente degli «oggetti». Di solito si tratta solo di riquadri contenenti del testo o di immagini, ma i programmi più sofisticati possono consentire l'inserzione di altri tipi di oggetti, purché abbiano una qualche rappresentazione visuale.

59.6.1 Scorrimento

Quando si inserisce un oggetto in un documento, occorre decidere in che modo il testo deve scorrere in prossimità di questo. La cosa più semplice, concettualmente, anche se le situazioni pratiche in cui torna utile sono poche, è quella dell'inserimento nel testo stesso, come se si trattasse di un carattere. In tal caso, l'oggetto si sposta assieme allo scorrere del testo. Ovviamente, un'inserzione del genere è utile solo quando si tratta di oggetti che occupano uno spazio molto piccolo, soprattutto dove l'altezza non è tale da disturbare in confronto all'altezza normale del testo in cui si inserisce.

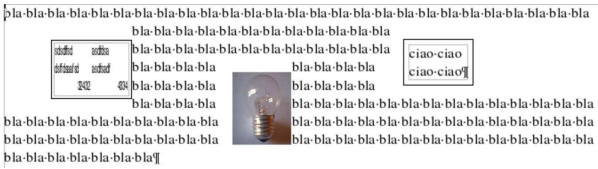
Figura 59.76. Una piccola immagine inserita come se fosse un carattere.

Un'altra modalità di scorrimento molto semplice è quella separata dall'oggetto, per cui il riquadro dell'oggetto occupa uno spazio orizzontale tutto suo e il testo non gli si affianca.

Figura 59.77. Scorrimento separato del testo rispetto all'oggetto.

Le modalità più comuni per lo scorrimento del testo rispetto a un oggetto sono quelle che lo fanno passare a fianco, distinguendo se questo scorrimento debba avvenire a sinistra, a destra o da entrambi i lati dell'oggetto. Inoltre, il testo potrebbe sovrapporsi all'oggetto, come se questo facesse parte di uno sfondo.

Figura 59.78. Si vedono tre oggetti: quello che appare alla sinistra prevede uno scorrimento del testo sulla destra; quello al centro prevede uno scorrimento da entrambi i lati; quello a destra prevede uno scorrimento del testo a sinistra.



Come si vede dalla figura, quando un oggetto prevede lo scorrimento da un lato solo, dall'altro lato, anche se non c'è nulla, rimane dello spazio vuoto. Quando si sposta l'oggetto manualmente, ma si desidera sempre che il testo scorra da uno dei lati, può essere conveniente lasciare che la scelta di questo lato avvenga dinamicamente (automaticamente).

Figura 59.79. AbiWord: *Formato*, *Immagine*; dettaglio della scelta del tipo di scorrimento del testo attorno all'immagine.

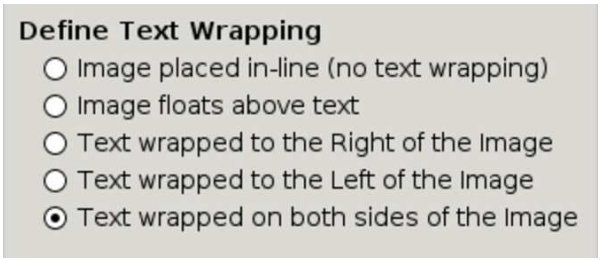
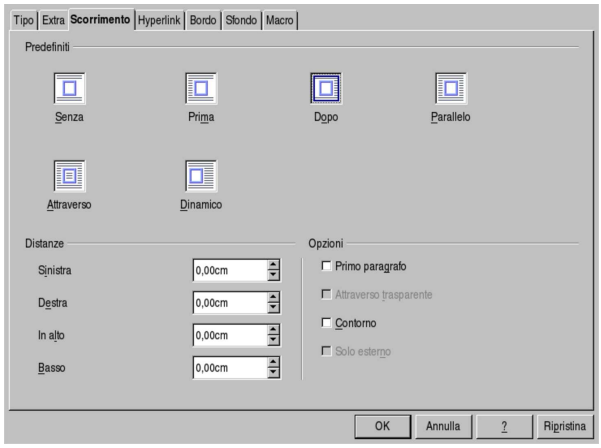


Figura 59.80. LibreOffice Writer: *Formato, Cornice, Scorrimento*, oppure *Formato, Immagine, Scorrimento*, a seconda del tipo di oggetto.



59.6.2 Ancoraggio

Quando si inseriscono degli oggetti in un documento, occorre stabilire in che modo questi vengono fissati allo stesso. Di solito si distinguono tre casi: l'ancoraggio alla pagina, al paragrafo o a un certo carattere. In pratica, a seconda di ciò che si prende come riferimento, l'immagine potrebbe spostarsi per adattarsi alle variazioni che si apportano al documento.

LibreOffice Writer considera in modo particolare l'ancoraggio di un'immagine come carattere. In pratica, quanto descritto a questo proposito nella sezione dedicata allo scorrimento, viene trattato da LibreOffice come parte del problema dell'ancoraggio.

Figura 59.81. AbiWord: *Formato*, *Immagine*; dettaglio della scelta del tipo di ancoraggio dell'immagine.

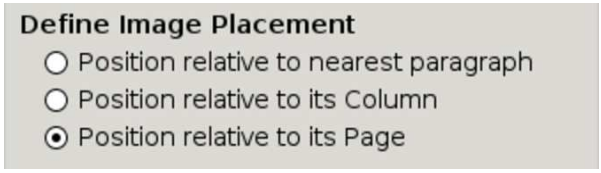
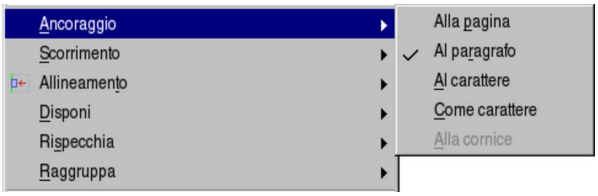


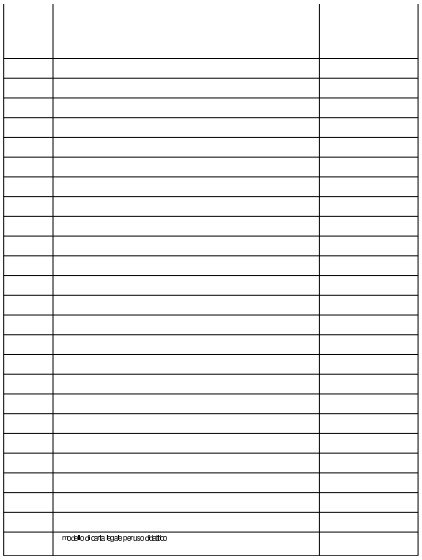
Figura 59.82. LibreOffice Writer: *Formato, Ancoraggio*.



59.7 Carta legale

Alcuni tipi di documento vanno redatti utilizzando della carta speciale, predisposta appositamente con margini e righe segnati. In questi casi si dice che il tale documento deve essere redatto «in carta legale». I modelli dei fogli adatti per la scrittura di documenti di questo tipo possono essere predisposti per la scrittura manuale o a macchina da scrivere, oppure per l'inserimento in una stampante. A seconda dei casi, i margini esatti possono variare leggermente, così come la spaziatura tra le righe, pertanto, a seconda della carta disponibile effettivamente, occorre adattare le misure del proprio documento elettronico.

Figura 59.83. Aspetto del modello didattico di carta «uso bollo», da usare per le esercitazioni.



59.7.1 Modello didattico

Per facilitare le esercitazioni di scrittura di documenti destinati alla stampa su carta legale, viene proposto un modello stampabile per un foglio A4, che simula la carta di questo tipo. Questo modello va stampato con la stessa stampante che si usa poi per stampare le esercitazioni, perché ogni stampante introduce un piccolo sfasamento dell'immagine, che così rimarrebbe uniforme.

Figura 59.84. Sorgente del modello didattico di carta «uso bollo», da usare per le esercitazioni.

```
%!PS-Adobe-2.0
%%DocumentPaperSizes: a4
%%EndComments
%%EndProlog
%
% Modello di foglio per bollato per la carta A4.
%
%%Page: 1 1
%
% Definizione dell'unità «cm».
%
/cm { 28.34645 mul } bind def
%
% Definizione dell'utilizzo dell'unità «cm».
%
1 cm dup scale
%
% Spessore delle linee (1/50 di cm, pari a 0,2 mm).
%
1 50 div setlinewidth
%
% Dichiara le intenzioni.
%
/Helvetica findfont 00.30 scalefont setfont
03.00  00.80  moveto
(modello di carta legale per uso didattico) show
%
% Traccia le linee verticali dei margini.
%
02.50  00.00  moveto  02.50  29.70  lineto stroke
16.00  00.00  moveto  16.00  29.70  lineto stroke
%
% Traccia le linee orizzontali.
%
00.00  01.20  moveto  21.00  01.20  lineto stroke
00.00  02.20  moveto  21.00  02.20  lineto stroke
00.00  03.20  moveto  21.00  03.20  lineto stroke
00.00  04.20  moveto  21.00  04.20  lineto stroke
00.00  05.20  moveto  21.00  05.20  lineto stroke
00.00  06.20  moveto  21.00  06.20  lineto stroke
00.00  07.20  moveto  21.00  07.20  lineto stroke
00.00  08.20  moveto  21.00  08.20  lineto stroke
00.00  09.20  moveto  21.00  09.20  lineto stroke
00.00  10.20  moveto  21.00  10.20  lineto stroke
00.00  11.20  moveto  21.00  11.20  lineto stroke
00.00  12.20  moveto  21.00  12.20  lineto stroke
00.00  13.20  moveto  21.00  13.20  lineto stroke
00.00  14.20  moveto  21.00  14.20  lineto stroke
00.00  15.20  moveto  21.00  15.20  lineto stroke
00.00  16.20  moveto  21.00  16.20  lineto stroke
00.00  17.20  moveto  21.00  17.20  lineto stroke
00.00  18.20  moveto  21.00  18.20  lineto stroke
00.00  19.20  moveto  21.00  19.20  lineto stroke
00.00  20.20  moveto  21.00  20.20  lineto stroke
00.00  21.20  moveto  21.00  21.20  lineto stroke
00.00  22.20  moveto  21.00  22.20  lineto stroke
00.00  23.20  moveto  21.00  23.20  lineto stroke
00.00  24.20  moveto  21.00  24.20  lineto stroke
00.00  25.20  moveto  21.00  25.20  lineto stroke
%
showpage
%%Trailer
%%EOF
```


59.7.2 Impaginazione

In base al modello presentato nella sezione precedente, l'impaginazione del testo richiede le caratteristiche seguenti:

Parametro	Valore
margine sinistro	2,70 cm
margine destro	5,30 cm
margine superiore	3,50 cm
margine inferiore	1,00 cm
interlinea fissa	1,00 cm

Nel caso di misure da esprimere in punti tipografici (PostScript), i valori si convertono nel modo seguente:

Parametro	Valore
margine sinistro	76,54 punti
margine destro	150,24 punti
margine superiore	99,21 punti
margine inferiore	28,35 punti
interlinea fissa	28,35 punti

59.7.3 Esercizio

Si scriva la richiesta seguente, seguendo il più possibile l'esempio. Il carattere tipografico utilizzato qui è precisamente il «FreeMono», con corpo di 12 punti tipografici (PostScript), ma in mancanza di questo carattere particolare, se ne può usare un altro di tipo dattilografico (a spaziatura uniforme), con un corpo simile.

Al Magnifico Rettore dell'università degli studi,

via Belenzani, 12, 38100 Trento

La sottoscritta, Martina Calpurnia, nata a Treviso il 12 dicembre 1969, residente a Ponzano Veneto (TV) C.A.P. 31050, in via delle sorgenti n. 1, avendo conseguito l'abilitazione all'esercizio della professione di dottore commercialista nella prima sessione dell'anno 2002

chiede

il rilascio del certificato sostitutivo del diploma di abilitazione all'esercizio della professione di dottore commercialista.

Trento, 12 dicembre 2002

Martina Calpurnia

modello di carta legale per uso didattico

Come si vede, dal momento che il testo deve adeguarsi ai margini orizzontali, viene usato un allineamento simultaneo a sinistra e a destra, con l'introduzione automatica degli spazi necessari tra le parole. Evidentemente, la parola '**chiede**' viene centrata e resa con un carattere più scuro, mentre il nome in calce della scrivente, appare allineato a destra.

Per accertare la correttezza dell'impaginazione, il documento va stampato mettendo nella stampante la carta preparata in precedenza con il modello mostrato all'inizio del capitolo.

59.7.4 Esercizio

«

Si scriva il testo seguente, simulando la forma richiesta per la scrittura in carta legale. Il testo può scorrere anche in modo leggermente differente, purché la forma rimanga sostanzialmente la stessa. La stampa va fatta usando la carta predisposta con il modello presentato precedentemente. Si producano due versioni, una con carattere dattilografico, l'altra con un carattere a spaziatura proporzionale, sobrio, con un corpo un po' più grande.

Alla Provincia di Cuneo, settore provinciale agricoltura, corso Dante 19, 12100 Cuneo

Il sottoscritto Tizio Tizi, nato a Cuneo il 01 gennaio 1980, residente in Cuneo, via tartufaia 11, telefono 123456789

chiede

a codesto settore di ottenere il rinnovo del tesserino di idoneità alla raccolta dei tartufi numero 123456789, rilasciato il 01 gennaio 2012.

Cuneo, 01 gennaio 2013

Tizio Tizi

Alla Provincia di Cuneo, settore provinciale agricoltura, corso Dante 19, 12100 Cuneo

Il sottoscritto Tizio Tizi, nato a Cuneo il 01 gennaio 1980, residente in Cuneo, via tartufaia 11, telefono 123456789

chiede

a codesto settore di ottenere il rinnovo del tesserino di idoneità alla raccolta dei tartufi numero 123456789, rilasciato il 01 gennaio 2012.

Cuneo, 01 gennaio 2013

Tizio Tizi

59.8 Stili e indici

«

Le caratteristiche legate al paragrafo, al testo, ma a volte anche ad altri componenti di un documento, possono essere uniformate attraverso la definizione di ***stili***. Ogni stile ha un nome e a questo vengono associate delle caratteristiche, a cui si vuole fare riferimento in modo complessivo. Quando un «oggetto» del documento viene associato a uno stile, le caratteristiche di quell'oggetto vengono conformate a tutte quelle dello stile relativo; quando poi le caratteristiche di un certo stile vengono modificate, tutti gli oggetti che vi sono associati ereditano simultaneamente questi cambiamenti.

Figura 59.90. Associazione di tre stili ad altrettanti componenti tipi di componenti del documento.

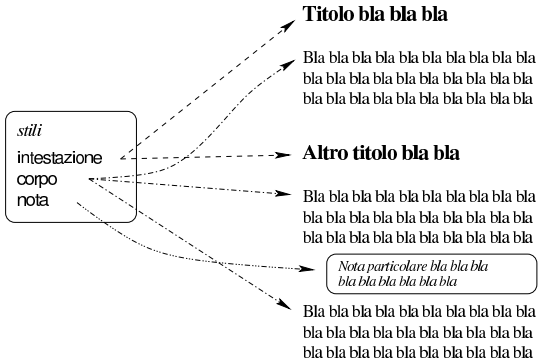
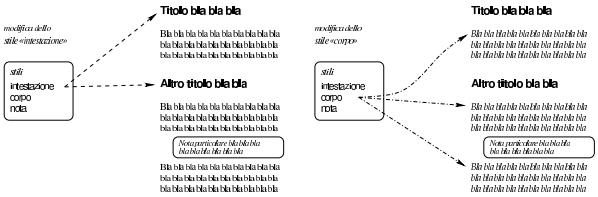


Figura 59.91. Modifica di uno stile e adattamento simultaneo di tutti i componenti che vi sono associati.

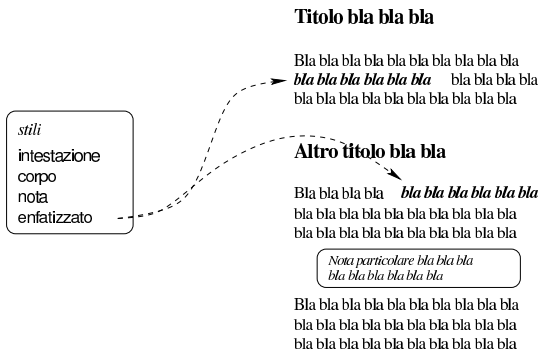


59.8.1 Competenze

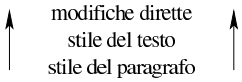
«

Gli stili dei programmi di scrittura visuale sono associati a un certo contesto e generalmente si considerano almeno il paragrafo e il testo lineare. Uno stile riferito al paragrafo riguarda un paragrafo nel suo insieme e può definire anche le caratteristiche del carattere, ma di tutto il paragrafo; uno stile riferito al testo lineare riguarda una porzione di testo, come eccezione rispetto alle caratteristiche generali del paragrafo.

Figura 59.92. Stile del carattere applicato come eccezione dello stile complessivo del paragrafo.



Se si applicano modifiche al paragrafo o al testo in modo diretto, senza l'uso di uno stile particolare, queste vengono considerate come un'eccezione ulteriore. In pratica si crea una sovrapposizione di definizioni, dove quanto rimane in superficie è ciò che conta per la composizione finale.



A seconda del programma usato per la scrittura, l'attribuzione di uno stile può eliminare o conservare le modifiche dirette già apportate. Per esempio, l'attribuzione di uno stile riferito al testo potrebbe rimanere subordinato alla presenza di una modifica esplicita, già effettuata, su quella stessa porzione. Pertanto, **quando si intendono usare gli stili**, per evitare complicazioni, **è meglio evitare completamente la modifica diretta delle caratteristiche dei paragrafi e del testo, usando per questo esclusivamente gli stili**.

Il problema appena descritto riguarda particolarmente LibreOffice Writer, il quale però offre una funzione apposita per riportare il testo allo stato «normale», ovvero quello che prescriverebbe lo stile: *Formato, Formattazione standard*. Eventualmente si può ottenere anche con la combinazione di tasti [*Ctrl Maiuscole Spazio*].

Quando si devono sovrapporre stili riferiti al testo, rispetto a uno stile riferito al paragrafo, è sempre necessario applicare gli stili del testo dopo quello riferito al paragrafo, perché ci sono programmi che con l'attribuzione dello stile del paragrafo azzerano gli stili per il testo preesistenti. D'altro canto, per ottenere un buon risultato con il minimo impegno possibile, conviene limitare il più possibile l'uso di stili che modificano solo piccole porzioni di testo.

59.8.2 Stili predefiniti

I programmi di scrittura visuale che gestiscono gli stili, si presentano inizialmente con un insieme di stili predefinito, con nomi più o meno intuitivi. Generalmente è possibile attribuire lo stile selezionandolo da un menù che fa parte degli strumenti posti in evidenza.

Figura 59.94. AbiWord: Selezione di uno stile dal menù presente negli strumenti principali.

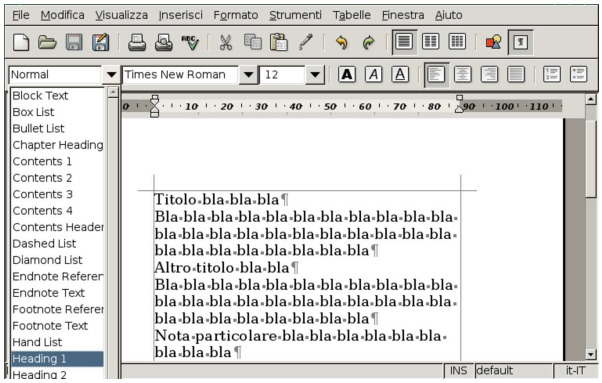
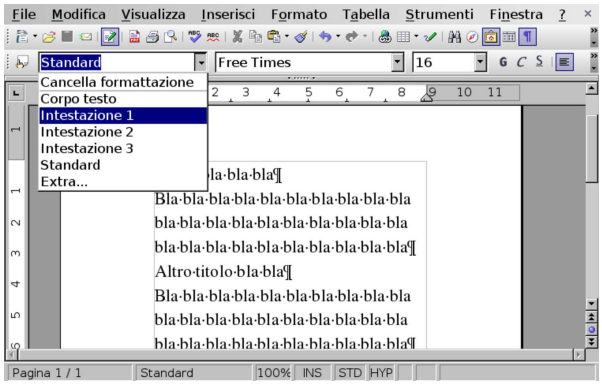


Figura 59.95. LibreOffice Writer: Selezione di uno stile dal menù presente negli strumenti principali.



Per ottenere un menù completo in una finestra separata, sempre disponibile, va richiamato lo «stilista». Nelle figure successive si vede Abiword e LibreOffice Writer, ognuno con il proprio stilista in evidenza.

Figura 59.96. AbiWord: *Strumenti*, *Stilista*. Per selezionare una voce dallo stilista, occorre un clic doppio con il primo tasto del mouse.

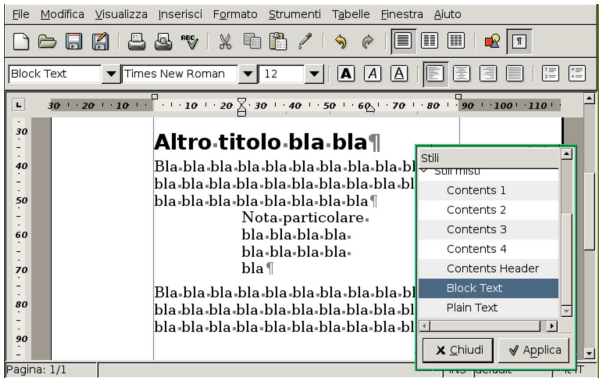
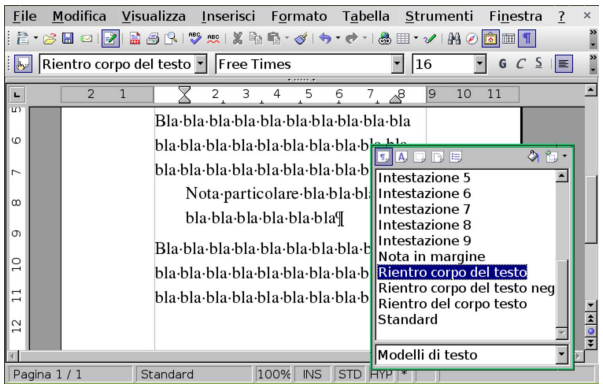


Figura 59.97. LibreOffice Writer: *Formato, Stilista*, ma si può utilizzare direttamente il tasto funzionale [*F11*], sia per far apparire lo stilista, sia per farlo scomparire. Per selezionare una voce dallo stilista, occorre un clic doppio con il primo tasto del mouse.



59.8.3 Modifica degli stili

Dalla breve descrizione fatta all'inizio del capitolo, si intende che l'utilità degli stili stia nella possibilità di modificarli e di far sì che tutti i componenti che vi sono associati ereditino immediatamente tali modifiche. A seconda del programma si accede alla modifica degli stili attraverso una voce del menù, oppure attraverso lo stilista stesso.

Figura 59.98. AbiWord: *Formato, Crea e modifica stili.*

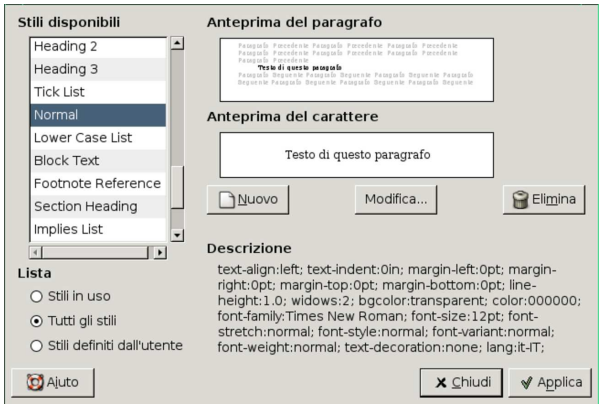


Figura 59.99. AbiWord: *Formato, Crea e modifica stili, Modifica*

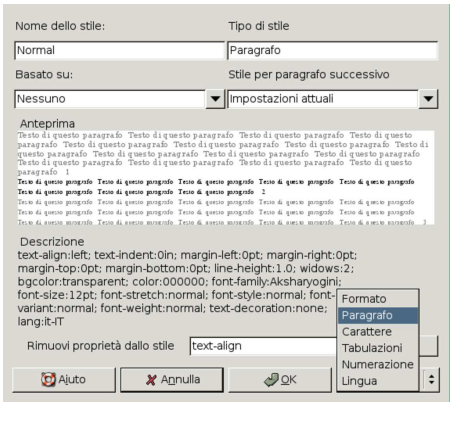


Figura 59.100. LibreOffice Writer: *Formato, Stilista*. Con il tasto destro del mouse si ottiene un menù contestuale, da dove si può procedere con la modifica di uno stile.

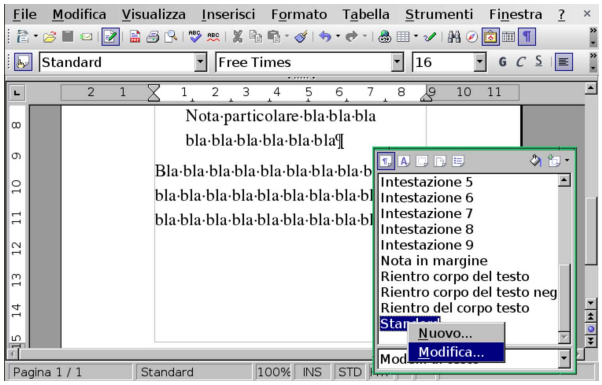
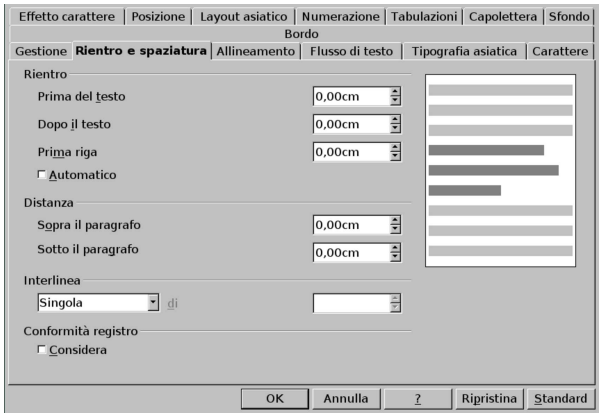


Figura 59.101. LibreOffice Writer: *Formato, Stilista, Modifica.* Si accede a una cartella di schede per la modifica dello stile.



59.8.4 Creazione di stili propri

Così come è concessa la modifica degli stili esistenti, è anche possibile aggiungerne degli altri. In generale ciò è utile quando non si comprende il significato originale di alcuni stili e non si trova una corrispondenza certa per ciò che si intende individuare. Per esempio si potrebbe volere uno stile per una nota evidenziata che non sembra avere uno stile predefinito appropriato, come si tenta invece di fare nelle figure successive.

Figura 59.102. AbiWord: *Formato, Crea e modifica stili, Nuovo.* Dopo l'indicazione del nome, si deve accedere al menù *Formato* per selezionare il contesto di caratteristiche da impostare.

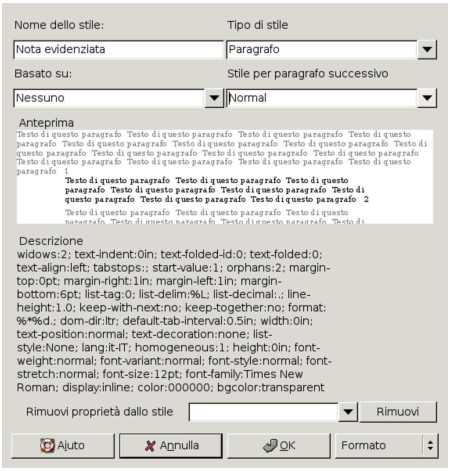


Figura 59.103. LibreOffice Writer: *Formato, Stilista*. Con il tasto destro del mouse si ottiene un menù contestuale, da dove si può procedere con la creazione di uno stile nuovo.

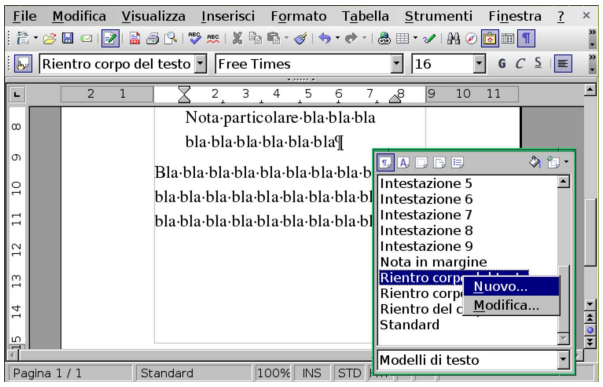
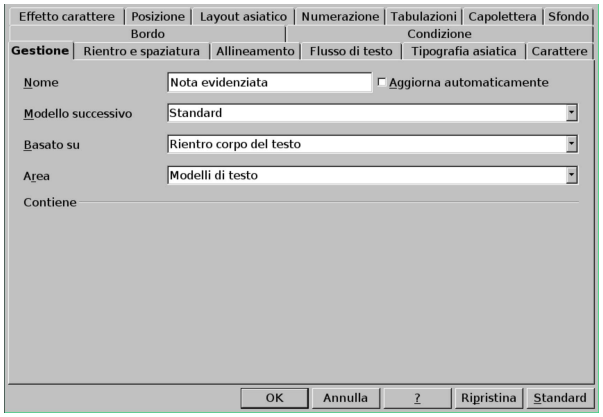


Figura 59.104. LibreOffice Writer: *Formato, Stilista, Nuovo*. Qui, cominciando dalla scheda *Gestione*, si procede per la creazione dello stile nuovo.



Di norma, così come possono essere aggiunti degli stili personali, questi possono anche essere rimossi. Inoltre, va osservato che gli stili seguono il documento, pertanto la modifica di uno stile nel documento ***x*** non interferisce con lo stato dello stesso stile nel documento ***y***.

59.8.5 Stili di uso comune

Purtroppo manca una definizione uniforme degli stili predefiniti che deve avere un programma di scrittura generalizzato. In ogni caso, l'utilizzatore deve sapere distinguere almeno quelli che riguardano i titoli dei contenuti, perché da quelli dipende la creazione di un indice. Lo specchietto successivo riepiloga i nomi degli stili minimi da utilizzare con Abiword e con LibreOffice Writer:

Contesto	Stile di Abiword	Stile di LibreOffice Writer
titolo più importante	`Heading 1` `Numbered Heading 1`	`Intestazione 1`
titolo di secondo livello	`Heading 2` `Numbered Heading 2`	`Intestazione 2`
titolo di terzo livello	`Heading 3` `Numbered Heading 3`	`Intestazione 3`
corpo normale	`Normal`	`Corpo testo` `Standard`

59.8.6 Indice generale

Se si utilizzano gli stili appropriati per i titoli delle varie sezioni del proprio documento, di norma è possibile ottenere un indice generale automatico. Per questo si deve posizionare il cursore del testo dove si vuole inserire l'indice, quindi si procede con i comandi necessari per generarlo.

Figura 59.106. AbiWord: *Inserisci, Indice*. Dal menù si ottiene istantaneamente l'indice, nella posizione in cui si trovava il cursore del testo. Successivamente si può passare alla configurazione dell'indice.

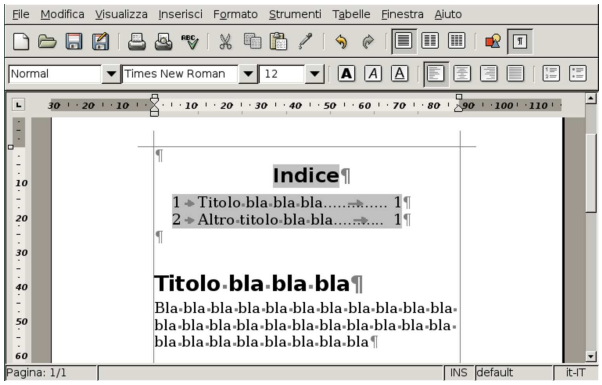


Figura 59.107. AbiWord: *Formato, Indice*. Avendo selezionato il testo dell'indice, dal menù si può richiedere la modifica delle sue caratteristiche, come si vede in questa figura.

Figura 59.108. LibreOffice Writer: *Inserisci, Indici, Indici*.

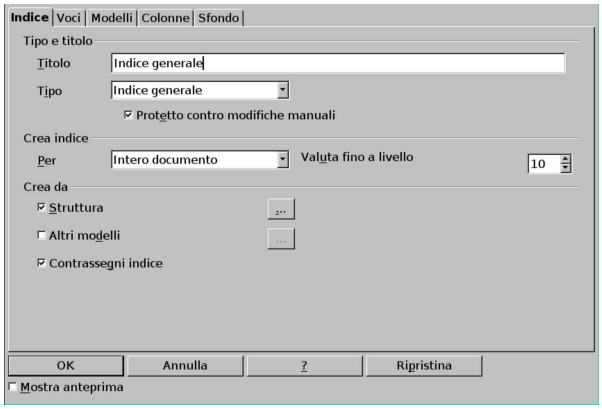
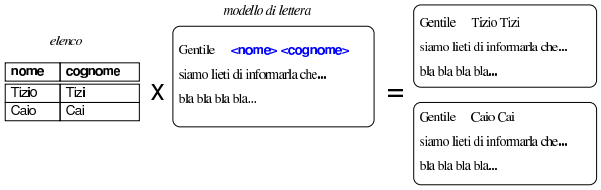


Figura 59.109. LibreOffice Writer: menù contestuale che si ottiene premendo il tasto destro del mouse sulle componenti dell'indice. Va osservata in particolare la voce che richiede l'aggiornamento automatico dell'elenco.

59.9 Stampa in serie o «stampa unione»

Per ***stampa in serie*** si intende l'associazione di un elenco di dati a un documento contenente delle variabili, in modo da produrre la riproduzione dello stesso documento per ogni voce dell'elenco, dove le variabili vengono sostituite con i campi corrispondenti dell'elenco. Il concetto è quello che si può vedere nell'esempio seguente, dove da un elenco di due nominativi e un modello di lettera, si vogliono ottenere due lettere personalizzate:



Generalmente, questi elenchi di dati possono essere redatti in vari modi, soprattutto con l'ausilio di un foglio elettronico o di una base di dati. Nel caso di LibreOffice Writer, si richiede precisamente l'uso di una base di dati secondo il proprio formato, ma in questo capitolo viene mostrato il procedimento per ottenere l'acquisizione di un elenco contenuto in un foglio elettronico, attraverso un esempio completo.

59.9.1 Preparazione di un elenco di indirizzi

A titolo di esempio, si parte da un file realizzato con LibreOffice Calc (il foglio elettronico di LibreOffice), contenente gli indirizzi, secondo lo schema seguente. Si può osservare che il nome dell'intestazione per la colonna contenente la città è stato scritto senza accento, per evitare complicazioni, ma il contenuto delle righe successive all'intestazione viene scritto regolarmente, usando le lettere accentate dove necessario.

	A	B	C	D	E
1	**nome**	**cognome**	**indirizzo**	**cap**	**citta**
2	Tizio	Tizi	via del semiconduttore, 1	31100	Treviso
3	Caio	Cai	via dell'integrato, 23	31050	Ponzano Veneto
4	Mevio	Mevi	via della CPU, 45	31050	Ponzano Veneto

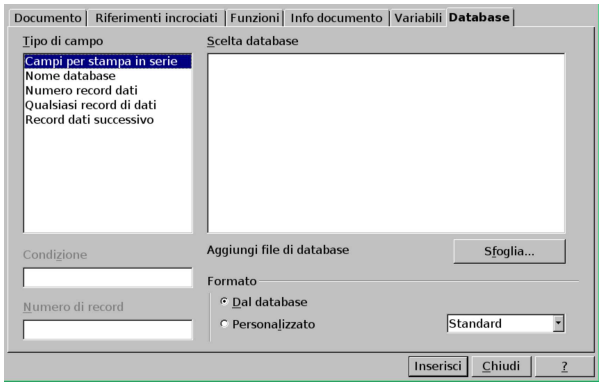
Si suppone di salvare questo file con il nome '`indirizzi.ods`', nella propria directory personale.

59.9.2 Acquisizione degli indirizzi

Attraverso LibreOffice Writer si acquisisce il file preparato con il foglio elettronico, in modo che diventi parte delle «sorgenti di dati». Questo procedimento implica la conversione automatica del file '`indirizzi.ods`' in una base di dati, rappresentata da un file con estensione '`.odb`'.

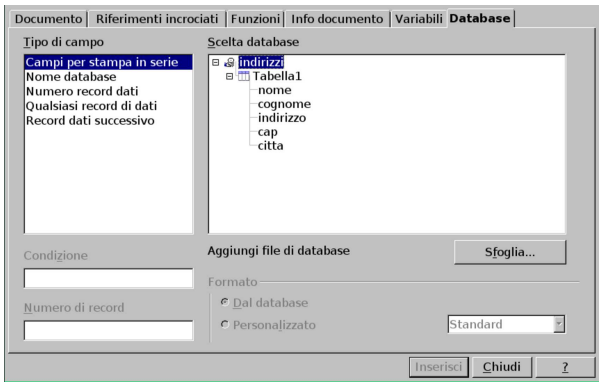
Si procede selezionando dal menù: *Inserisci*, *Comando di campo*, *Altro*. Si ottiene una finestra contenente delle cartelle, dove si deve selezionare la scheda *Database* e nell'elenco del tipo va scelta la voce *Campi per stampa in serie*.

Figura 59.112. LibreOffice Writer: *Inserisci*, *Comando di campo*, *Altro*, *Database*.



Selezionando il pulsante *Sfoglia* si accede a una finestra da cui è possibile selezionare il file contenente gli indirizzi ('`indirizzi.ods`'); fatto questo, il file viene trasformato automaticamente in una base di dati (costituita probabilmente dal file '`indirizzi0.odb`') e segnalato, come si vede nella figura successiva.

Figura 59.113. LibreOffice Writer: *Inserisci*, *Comando di campo*, *Altro*, *Database*, dopo la selezione di un file.

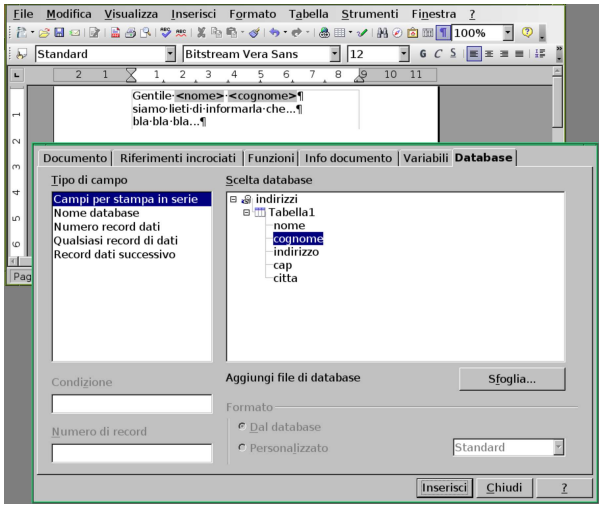


A questo punto si può mantenere aperta la finestra contenente l'elenco dei campi per la stampa in serie, mentre si procede a scrivere il testo della lettera.

59.9.3 Inserimento dei campi

Durante la scrittura della lettera, quando ciò è appropriato, si possono inserire i campi provenienti dall'elenco di indirizzi, mettendo in evidenza il nome del campo nella finestra che li contiene e selezionando il pulsante grafico *Inserisci*.

Figura 59.114. LibreOffice Writer: inserimento dei campi nel testo di una lettera.

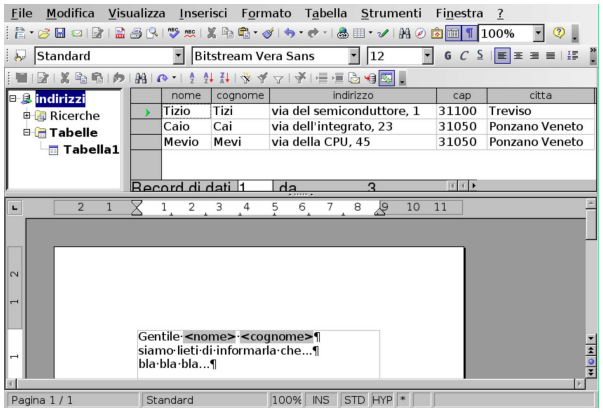


Nel testo del documento, i campi variabili appaiono evidenziati e portandovi sopra il puntatore del mouse si vede precisamente la loro origine. Terminata la selezione dei campi, la finestra contenente l'elenco dei campi per la stampa in serie può essere chiusa.

59.9.4 Visualizzazione dei dati

Una volta terminata la selezione dei campi e chiusa la finestra contenente l'elenco dei campi per la stampa in serie, conviene mettere in evidenza i dati collegati al proprio documento. Per questo si seleziona dal menù la voce *Visualizza, Sorgente dati*.

Figura 59.115. LibreOffice Writer: *Visualizza, Sorgente dati.*



Avendo verificato che i dati a cui fa riferimento il proprio documento sono quelli corretti, si può passare alla stampa in serie.

59.9.5 Stampa in serie

Attraverso le figure successive, viene mostrato il procedimento per arrivare alla stampa in serie del documento, in modo da ottenere una copia, adattata, per ogni voce dell'elenco di indirizzi. Si inizia selezionando la voce *Strumenti, Stampa guidata in serie*. Se la stampa riguarda il documento attivo e l'elenco di voci associate, nella maggior parte dei casi è sufficiente confermare e passare alle fasi successive.

Figura 59.116. LibreOffice Writer: *Strumenti, Stampa guidata in serie*, prima fase.

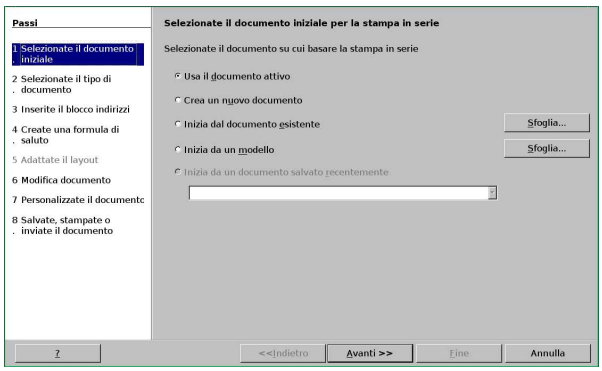


Figura 59.117. LibreOffice Writer: *Strumenti, Stampa guidata in serie*, seconda fase.

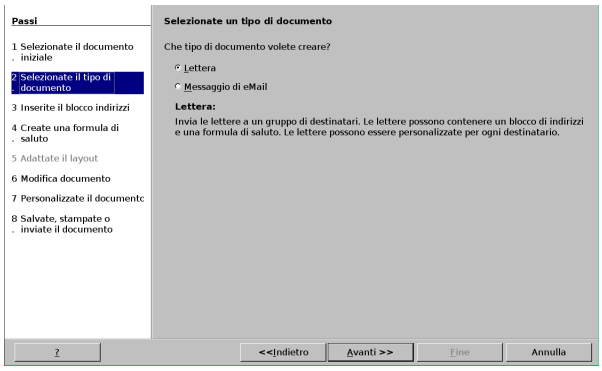


Figura 59.118. LibreOffice Writer: *Strumenti, Stampa guidata in serie*, terza fase.

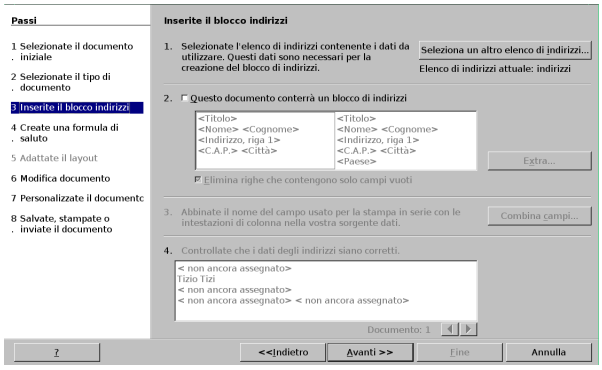
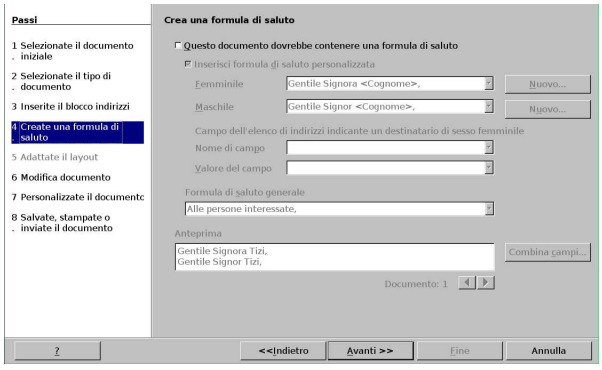


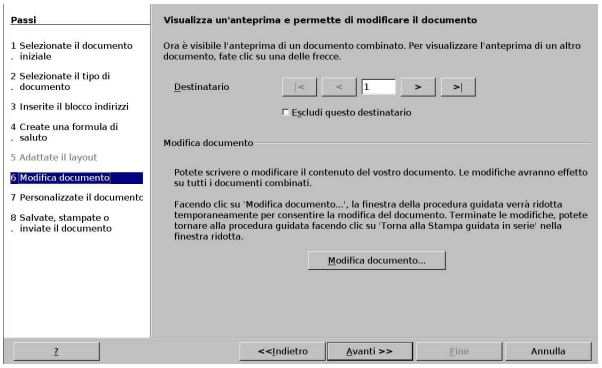
Figura 59.119. LibreOffice Writer: *Strumenti, Stampa guidata in serie*, quarta fase.



Non avendo fatto cambiamenti, la quinta fase viene saltata, perché

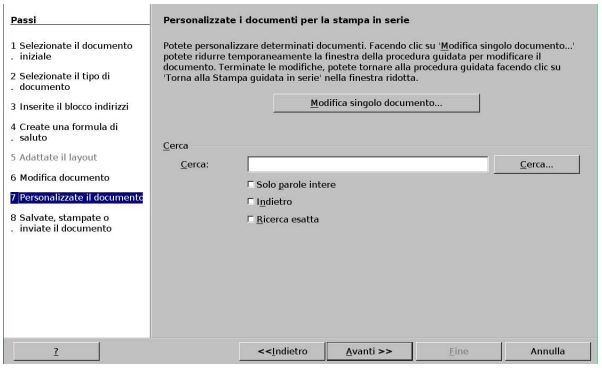
risulta superflua.

Figura 59.120. LibreOffice Writer: *Strumenti*, *Stampa guidata in serie*, sesta fase.



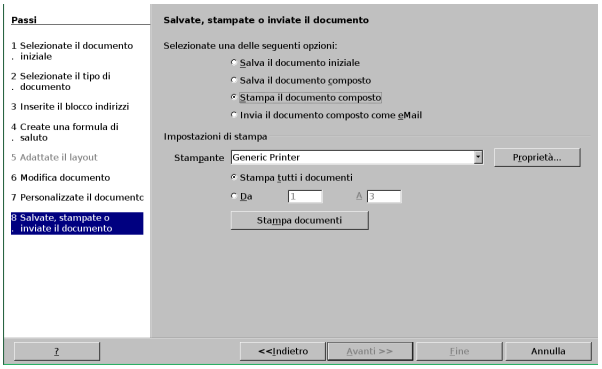
Nel passaggio dalla sesta alla settima fase, viene prodotto un documento temporaneo, composto dalla ripetizione del documento originale, sostituendo ogni volta i campi variabili con i dati relativi dell'elenco di indirizzi.

Figura 59.121. LibreOffice Writer: *Strumenti*, *Stampa guidata in serie*, settima fase.



Nell'ultima fase va specificato cosa si vuole fare con il documento composto ottenuto nel passaggio precedente. In questo caso si seleziona la stampa, ma potrebbe anche essere salvato.

Figura 59.122. LibreOffice Writer: *Strumenti*, *Stampa guidata in serie*, ottava fase.



In ogni caso, dopo la stampa, il documento composto rimane aperto e si può disporre di questo come si vuole.

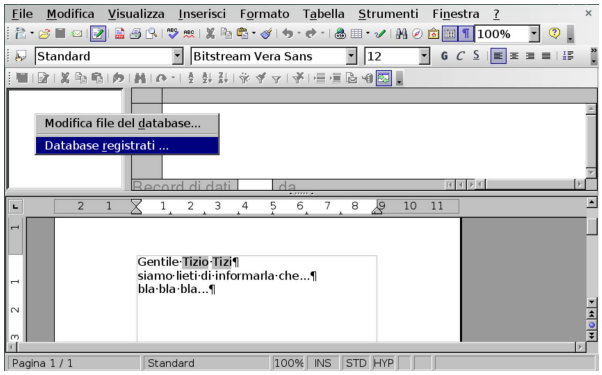
59.9.6 Associazione di un elenco alternativo

«

Il file contenente il documento, contiene delle «variabili» associate a campi di una tabella di una base di dati. Nel caso dell'esempio descritto nelle sezioni precedenti, la variabile del nome dovrebbe risultare associata a '**`indirizzi.Tabella1.nome`**', mentre quella del cognome a '**`indirizzi.Tabella1.cognome`**'. Quando si riapre un documento con queste variabili, occorre ripristinare il collegamento con i dati a cui si riferiscono, altrimenti la stampa in serie non può avere luogo. D'altra parte, va anche considerata la possibilità di associare a un certo documento un elenco differente di voci, pur mantenendo la corrispondenza nei nomi dei campi.

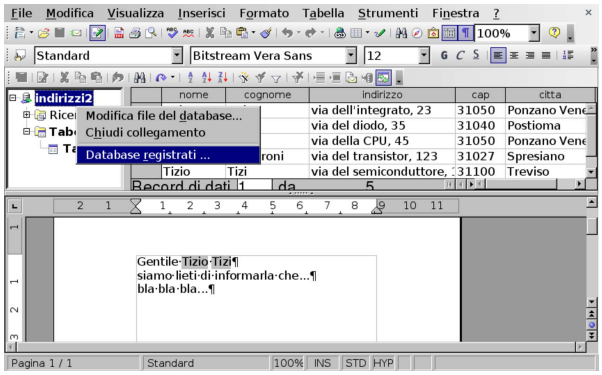
Dal riquadro che si ottiene con la visualizzazione della sorgente dei dati (*Visualizza*, *Sorgente dati*), dove dovrebbe apparire l'elenco di tali basi di dati, premendo il tasto destro del mouse si ottiene un menù contestuale, dal quale è possibile eliminare o selezionare i file di proprio interesse. Va osservato che questo elenco viene aggiornato automaticamente se si acquisiscono dati attraverso la procedura *Inserisci*, *Comando di campo*, *Altro*, *Database*,... A ogni modo, attraverso la voce *Database registrati* del menù contestuale che si vede anche nella figura successiva, conviene fare pulizia delle basi di dati a cui non si è interessati.

Figura 59.123. LibreOffice Writer: *Visualizza*, *Sorgente dati*. Premendo con il tasto destro del mouse si ottiene un menù contestuale per modificare l'elenco delle basi di dati.



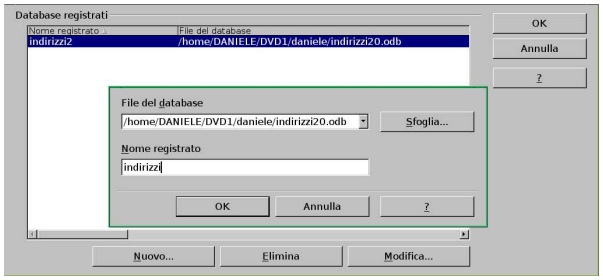
Si suppone di acquisire in qualche modo la base di dati '**`indirizzi2`**', compatibile con il documento realizzato in precedenza, ma con un nome differente.

Figura 59.124. LibreOffice Writer: *Visualizza*, *Sorgente dati*. La base di dati '**`indirizzi2`**' ha un nome incompatibile con le variabili già inserite nel documento.



Si passa quindi alla modifica del nome, attraverso la voce *Database registrati*, come si vede nella figura successiva. Quando il nome della base di dati che appare nel riquadro della sorgente dati corrisponde a quello annotato nelle variabili del documento, si può procedere come già descritto per la stampa in serie.

Figura 59.125. LibreOffice Writer: modifica del nome della base di dati.



[1] **Gucharmap** GNU GPL

Lavagna elettronica

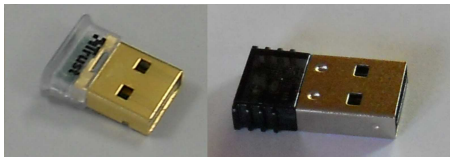
Per «lavagna elettronica», o «lavagna interattiva», si intende generalmente una superficie sulla quale si proietta la schermata di un elaboratore, dove è però possibile interagire con l'elaboratore stesso, attraverso il tatto (in quanto la superficie è sensibile) o degli strumenti appositi. In questo capitolo si considerano le soluzioni economiche per la realizzazione di una tale tipo di lavagna, senza bisogno di superfici sensibili (e costose).

Dal momento che questo tipo di strumento è destinato prevalentemente all'uso nella scuola, dato che in tale ambiente ci si trova spesso nell'impossibilità pratica di poter scegliere il sistema operativo, in questo capitolo si mostra sia il software adatto per sistemi GNU/Linux, sia quello per MS-Windows.

60.1 Bluetooth con i sistemi GNU/Linux

Gli strumenti che si usano generalmente per rendere interattiva una superficie di proiezione, senza l'ausilio di grandi risorse economiche, si connettono di norma via radio attraverso la tecnologia Bluetooth. Ciò richiede l'utilizzo di un adattatore Bluetooth nell'elaboratore, il quale può essere incorporato o connesso su una porta USB.

Figura 60.1. Adattatori Bluetooth USB: a sinistra un modello Trust 16008; a destra un Sitecom CN-523.



In un sistema GNU/Linux, la gestione dei dispositivi Bluetooth avviene attraverso funzioni del kernel, di solito in forma di moduli, che di norma vengono caricati automaticamente nel momento in cui i dispositivi risultano disponibili. Chi compila il kernel Linux in proprio, deve sapere che queste funzionalità risiedono nella voce *Network support, Bluetooth subsystem support.* La gestione dei protocolli dei dispositivi Bluetooth è invece compito di Bluez (*http://www.bluez.org* . Naturalmente, in un sistema GNU/Linux che prevede una gestione ordinata dei pacchetti applicativi e delle dipendenze, quando si installa del software che si avvale della connettività Bluetooth, si è naturalmente condotti a installare anche Bluez.

Del software che compone Bluez è importante conoscere almeno l'uso elementare di '**`hcitool`**', con il quale è possibile verificare il corretto riconoscimento di un adattatore Bluetooth e l'esistenza di una connessione in atto. Il comando seguente elenca i dispositivi Bluetooth:

```
$ hcitool dev [Invio]
```

```
Devices:
        hci0    00:02:72:AA:16:64
```

In questo caso si vede la presenza di un adattatore, a cui viene attribuito il nome '**hci0**', con l'indirizzo che si vede nell'esempio. Il comando seguente, invece, elenca i dispositivi remoti, alla portata del proprio adattatore:

```
Scanning ...
        00:21:FC:7E:48:E3       Clementina
        00:17:E6:7E:48:9D       Tino
        00:1D:F6:C8:15:05       SGH-E390
        8C:56:C5:D9:69:D3       Nintendo RVL-CNT-01
```

Con il comando seguente, si vedono le connessioni in corso, tra l'adattatore locale ed eventuali unità remote:

```
Connections:
        < ACL 8C:56:C5:D9:69:D3 handle 11 state 1 lm MASTER
```

Come si vede, la connessione riguarda il dispositivo remoto Nintendo, già apparso nell'elenco di quelli presenti.

60.2 Bluetooth con i sistemi MS-Windows

La gestione delle connessioni Bluetooth nei sistemi MS-Windows avviene tramite del software che accompagna generalmente l'adattatore hardware acquistato. Generalmente, già nella fase di installazione di questo software, viene proposto di scandire le unità che possono collegarsi via radio, secondo il protocollo Bluetooth, cosa che generalmente conviene assecondare. Nelle figure successive si vede la preparazione del software che accompagna un adattatore Sitecom CN-523.

Figura 60.5. In questa fase, si richiede di attivare i dispositivi remoti, in modo tale che questi si annuncino. Nel caso del Wiimote, si tratta di premere contemporaneamente i bottoni 1 e 2.

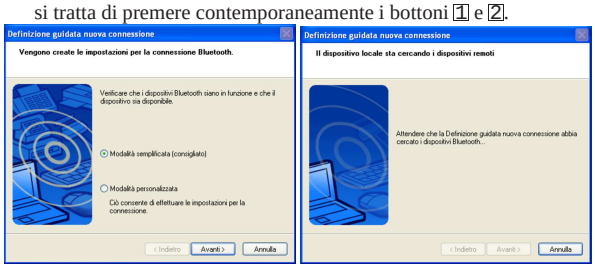


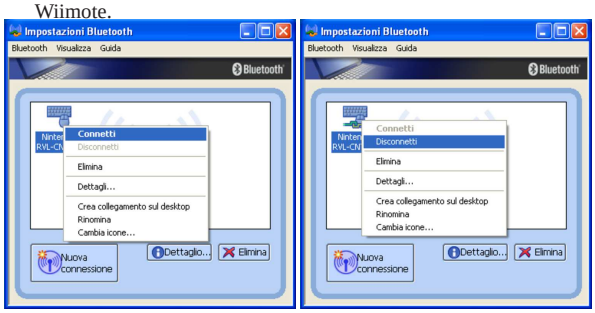
Figura 60.6. Qui viene individuato un dispositivo Wiimote, corrispondente alla sigla RVL-CNT-01. Questo dispositivo non richiede un codice PIN.

Figura 60.7. In questa fase può essere necessario intervenire nuovamente sul dispositivo remoto, benché già individuato, per fare sì che questo si annunci (con il Wiimote si tratta di premere contemporaneamente i bottoni 1 e 2).

Figura 60.8. Conclusione della registrazione del dispositivo remoto.

I dispositivi Bluetooth remoti, quando servono, vanno connessi e staccati espressamente (salva comunque la possibilità di spegnere tali dispositivi, interagendo con gli stessi). Pertanto, prima di poter usare uno strumento come il Wiimote, attraverso il software più appropriato, occorre comunque procedere alla sua connessione, ricordando poi di chiuderla alla fine dell'utilizzo. Nelle figure successive si vede la connessione di un Wiimote e la conclusione di questa attraverso il software dell'adattatore Sitecom CN-523.

Figura 60.9. Attivazione e conclusione di una connessione con il Wiimote.



60.2.1 Problemi con il PIN

Purtroppo, non tutti i software MS-Windows di gestione degli adattatori Bluetooth sono in grado di connettersi al Wiimote. Il problema può nascere dal fatto che il Wiimote non richiede l'uso di un codice PIN, mentre alcuni software pretendono che ci sia sempre. Ciò succede, per esempio, con il software allegato all'adattatore Trust 16008 che appare nella figura di esempio all'inizio del capitolo. Tuttavia, di norma tali adattatori funzionano perfettamente in un sistema GNU/Linux, dato che in quel caso non c'è bisogno di software esterni per la loro gestione, essendo questo un compito svolto interamente dal kernel.

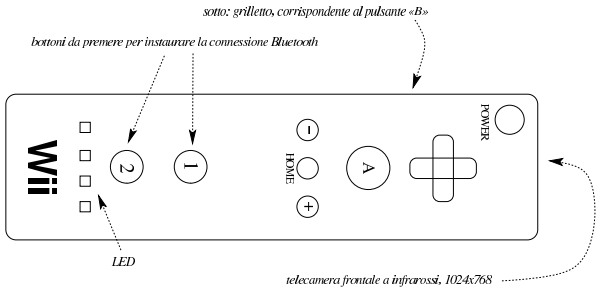
60.3 Wiimote

Il nome Wiimote si riferisce al telecomando Wii, della Nintendo, il quale incorpora dei sensori gravitazionali (accelerometri) e una telecamera a raggi infrarossi. Si tratta di un telecomando in quanto si collega a un elaboratore o a un'altra apparecchiatura, via radio, usando la tecnologia Bluetooth.

La telecamera a infrarossi è a fuoco fisso, con una risoluzione di 1024×768.

Quando si vuole instaurare il collegamento Bluetooth con l'elaboratore, occorre far riconoscere il dispositivo. Per questo, appena il software è pronto per questa fase, si richiede la pressione simultanea dei pulsanti 1 e 2, sul dispositivo Wiimote. In quel momento, mentre il Wiimote tenta di farsi individuare, lampeggiano quattro led sul dispositivo stesso. Se il Wiimote viene installato stabilmente, in una sua collocazione fissa, e viene alimentato dalla rete (con un alimentatore adatto), questi due pulsanti possono essere bloccati in modo da risultare sempre premuti; così, all'accensione il Wiimote si mette in cerca di una connessione, continuando fino a che questa si verifica effettivamente.

Figura 60.10. Telecomando Wii.



60.4 Wminput

In un sistema GNU/Linux, il programma Wminput[1] consente di abbinare delle funzioni al telecomando Wii. Nella sua configurazione predefinita, dovrebbe essere in grado di controllare le funzioni del mouse, tenendo in mano il Wiimote, muovendolo in alto, in basso e ruotandolo, facendo clic con il bottone A e B (il grilletto).

Per poter usare il programma Wminput, è necessario che sia stato caricato il modulo del kernel '**uinput**'; inoltre, l'eseguibile '**wminput**' deve funzionare con i privilegi dell'utente '**root**'; pertanto, se si intende utilizzarlo, può convenire il fatto di associargli i permessi SUID-root

```
# modprobe uinput [Invio]
```

```
# chown root /usr/bin/wminput [Invio]
```

```
# chmod u+s /usr/bin/wminput [Invio]
```

Quando si avvia il programma eseguibile '**wminput**', questo chiede inizialmente di mettere il Wiimote nella modalità che ne consente il riconoscimento, premendo simultaneamente i bottoni 1 e 2. Se il collegamento Bluetooth avviene con successo, si può interagire con il telecomando Wii come se fosse un mouse. Tuttavia, va chiarito che si tratta di un uso abbastanza faticoso e, per questo, poco consigliabile.

Se si usa Wminput per controllare il Wiimote, al termine dell'utilizzo è necessario chiudere il collegamento premendo a lungo il bottone di spegnimento sul dispositivo Wii.

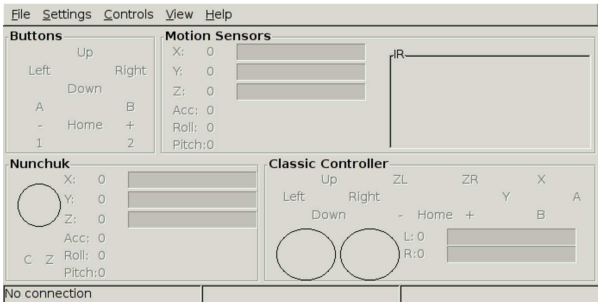
L'uso di Wminput non viene qui approfondito, ma va tenuto presente che se si intende utilizzarlo occorre documentarsi bene e sperimentare a lungo ciò a cui si è interessati; inoltre, va tenuto conto che questo programma, se lasciato in funzione, interferisce con programmi come Gtkwhiteboard, creando complicazioni ulteriori.

60.5 Wmgui

In un sistema GNU/Linux, il programma grafico Wmgui[2] è un monitor del funzionamento di un telecomando Wii; in modo particolare consente di individuare l'inquadratura della telecamera a raggi infrarossi. Wmgui non ha bisogno del modulo del kernel '**uinput**' e non ha bisogno di privilegi particolari per poter funzionare.

```
$ wmgui [Invio]
```


Figura 60.11. Wmgui in funzione.



Dal menù *File* si deve selezionare la voce *Connect*, per connettere il programma al dispositivo Wii, sul quale occorre premere simultaneamente i bottoni 1 e 2.

Figura 60.12. Menù di Wmgui: connessione.

Wmgui interagisce con il telecomando Wii in base a ciò che viene selezionato nel menù *Settings*. Per esempio, per verificare l'inquadratura della telecamera a infrarossi, occorre abilitare la voce *IR Data*.

Figura 60.13. Menù di Wmgui: impostazione.

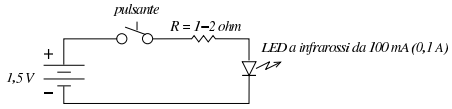
Quando è attiva la voce *IR Data*, è possibile, con l'aiuto di una penna a infrarossi, verificare il campo inquadrato dalla telecamera, osservando il punto che appare nel riquadro rettangolare, contrassegnato dalla sigla «IR» (*infrared*). Quando si vuole usare il Wiimote con un programma come Gtkwhiteboard, per piazzare e orientare correttamente il Wiimote è necessario avvalersi proprio di Wmgui in questo modo.

Quando si chiude correttamente il funzionamento di Wmgui, la connessione Bluetooth con il Wiimote viene conclusa regolarmente, senza bisogno di intervenire sul bottone di spegnimento del dispositivo. Eventualmente, la chiusura della connessione la si può richiedere espressamente con la voce *Disconnect* del menù *File*.

60.6 Penna a infrarossi

La modalità di utilizzo più semplice del Wiimote, per la gestione di una lavagna interattiva, è quella che sfrutta la telecamera a infrarossi, lasciando il dispositivo in una posizione fissa a inquadrare la superficie dello schermo di proiezione. In questo modo, il Wiimote registra il movimento di un segnale a infrarossi, generato da una «penna», avente sulla sommità un diodo LED a infrarossi. Di solito, questa penna deve essere realizzata artigianalmente.

Figura 60.14. Schema della penna a infrarossi.



Nella schema appena mostrato si intende usare una pila stilo da 1,5 V e un LED a infrarossi da 100 mA. Dal momento che ai capi del LED può formarsi una differenza di potenziale che va da 1,3 V a 1,4 V, la resistenza che controlla il flusso di corrente va da 2 ohm a 1 ohm rispettivamente. Per essere sicuri è necessario provare con uno strumento di misura preciso[3] il flusso di corrente con una resistenza da 2 ohm: tenendo conto che lo strumento introduce una piccola resistenza, se il flusso di corrente misurato risulta molto inferiore a 50 mA, si può tentare di usare una resistenza da 1 ohm (sperando di non bruciare il LED).

Per verificare il funzionamento della penna, dal momento che la radiazione infrarossa emessa dal LED non è visibile dall'occhio umano, ci si può avvalere di una fotocamera digitale, la quale invece evidenzia tale emissione.

Figura 60.15. L'emissione a infrarossi è visibile con l'ausilio di una fotocamera digitale.

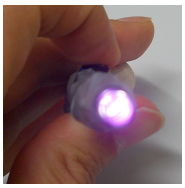
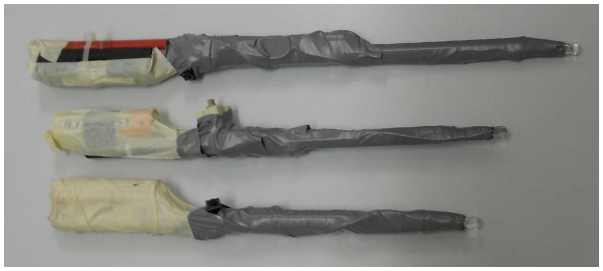


Figura 60.16. Penne a infrarossi fatte in casa.

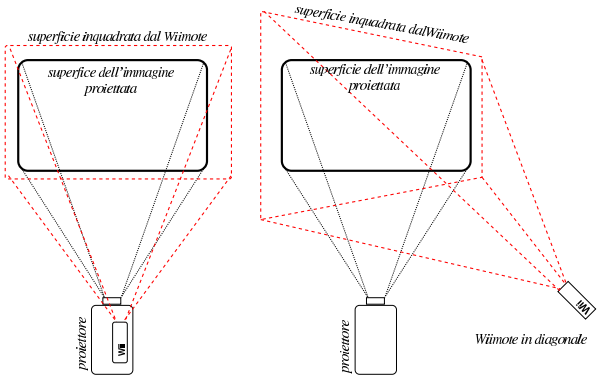


60.7 Inquadratura dello schermo

Se si intende usare il Wiimote per inquadrare la superficie di uno schermo, per renderlo interattivo attraverso una penna a infrarossi, è necessario collocare il dispositivo alla distanza giusta, orientandolo nel modo più appropriato. Lo schermo che diventa interattivo può essere qualunque cosa sia in grado di visualizzare ciò che appare o apparirebbe sul terminale grafico: generalmente si tratta di un pannello o di un telo sul quale si proietta questa immagine.

Considerando che il caso della proiezione sia quella più comune, ci sono due possibilità per collocare il Wiimote: vicino al proiettore stesso, oppure in una posizione obliqua, come si vede nella figura successiva.

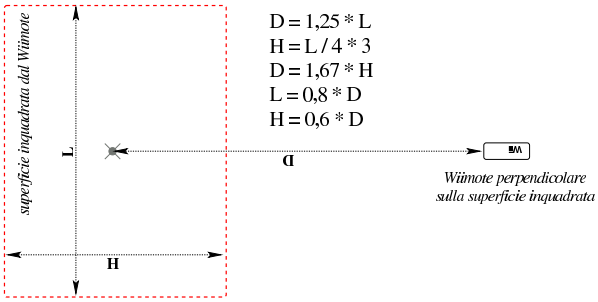
Figura 60.17. Inquadrature.



Wiimote all'incirca in perpendicolare

Quando si inquadra lo schermo con il Wiimote in modo perpendicolare, o quasi, come nella parte sinistra dello schema mostrato nella figura, la penna a infrarossi deve essere usata in modo obliquo, altrimenti il Wiimote non è in grado di leggere la luce a infrarosso emessa dalla punta appoggiata sulla superficie dello schermo. Quando invece l'inquadratura è diagonale, la penna deve essere usata in modo quasi perpendicolare sullo schermo.

Figura 60.18. Indicativamente, il rapporto tra la distanza del Wiimote dalla superficie inquadrata e la larghezza di tale superficie è di 1,25 (la larghezza inquadrata è il 125 % della distanza); inoltre, la superficie inquadrata è un rettangolo con rapporto 4/3.



Ogni inquadratura ha vantaggi e svantaggi che si possono intuire facilmente. Di sicuro, se il proiettore è mobile, è necessario fissare il Wiimote sullo stesso, dopo aver verificato qual è lo zoom ottimale del proiettore per fare in modo che questo sia compatibile con l'inquadratura fissa del Wiimote.

Va osservato che l'inquadratura del Wiimote deve essere leggermente più ampia di quella dell'immagine proiettata, facendo attenzione però a non esagerare, perché la risoluzione della telecamera a infrarossi è di soli 1024×768 punti, e non si può sprecare inutilmente.

Per fissare e orientare il Wiimote, è necessario un programma come Wmgui, in modo da riuscire a delimitare la superficie inquadrata dalla telecamera a infrarossi.

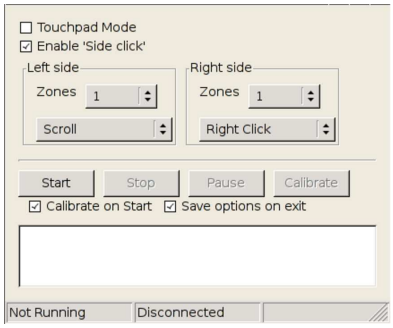
Figura 60.19. Fissaggio di fortuna del Wiimote sul proiettore.

60.8 Gtkwhiteboard

In un sistema GNU/Linux, il programma grafico Gtkwhiteboard[4] consente di utilizzare un telecomando Wii al posto del mouse, sfruttando la telecamera a infrarossi.

```
$ gtkwhiteboard [ Invio ]
```

Figura 60.20. Gtkwhiteboard, evidenziando la configurazione di funzionamento proposta.

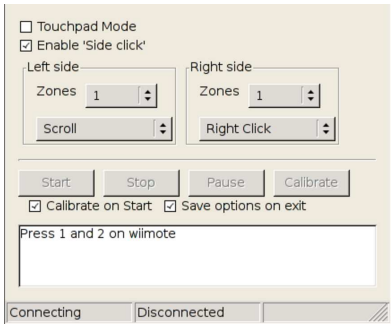


Nella figura si vede che risulta attiva la funzione *Side click*. In tal modo, è possibile individuare fino a sei aree, in cui fare un «clic» con la penna a infrarossi, esterne allo schermo usato per la proiezione, per ottenere delle funzioni speciali.

Figura 60.21. Le aree esterne alla zona circoscritta dalla proiezione.

Per attivare il collegamento tra Gtkwhiteboard e il telecomando Wii, occorre selezionare il bottone grafico START e premere simultaneamente i bottoni 1 e 2 sul Wiimote.

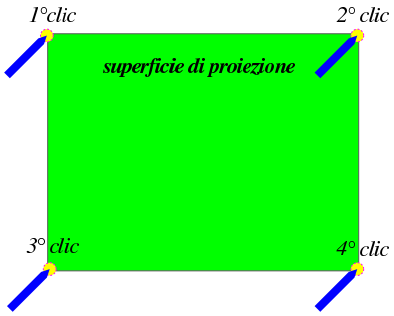
Figura 60.22. Gtkwhiteboard in attesa di individuare il dispositivo Wiimote.



Quando il Wiimote viene individuato, viene richiesto di fare «clic» con la penna a infrarossi, sugli angoli della proiezione; questa operazione corrisponde alla ***calibratura***.

A differenza di altri programmi simili, Gtkwhiteboard non mostra dei riferimenti sullo schermo, ma chiede semplicemente di fare un clic sull'angolo superiore sinistro, poi su quello superiore destro, poi su quello inferiore sinistro e infine su quello inferiore destro. Questa modalità si giustifica per poter conoscere esattamente i bordi dello schermo, allo scopo di poter sfruttare il bordo esterno, ammesso che sia inquadrato dal Wiimote, per delle funzioni speciali.

Figura 60.23. Calibratura dello schermo.



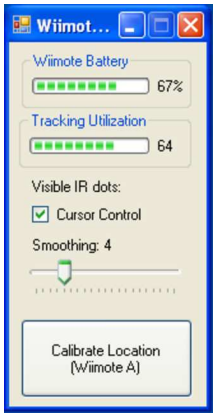
Al termine della calibratura è possibile usare la penna a infrarossi come se fosse un mouse, con il quale si può fare solo il clic con il tasto sinistro, senza poter trasmettere l'informazione sul movimento del puntatore, quando non si preme alcunché.

Al termine dell'utilizzo, si deve selezionare il bottone STOP, con cui viene anche spento il Wiimote.

60.9 WiimoteWhiteboard

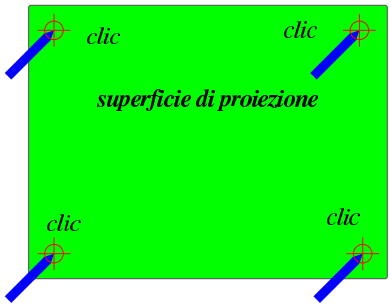
In un sistema MS-Windows, il programma grafico WiimoteWhiteboard[5] consente di utilizzare un telecomando Wii al posto del mouse, sfruttando la telecamera a infrarossi. A differenza di quanto avviene in un sistema GNU/Linux, la connessione con il Wiimote deve essere effettuata prima di avviare il programma, attraverso gli strumenti software dell'adattatore Bluetooth.

Figura 60.24. WiimoteWhiteboard.



Nella modalità di funzionamento evidenziata dalla figura appena apparsa, quando si seleziona la voce CALIBRATE LOCATION, oppure si preme il bottone A sul Wiimote, vengono mostrati dei «mirini» sullo schermo, al centro dei quali è necessario fare clic con la penna a infrarossi.

Figura 60.25. Calibratura dello schermo.



60.10 Uniboard (non libero)

In un sistema MS-Windows o in un sistema MacOSX, il programma Uniboard[6] consente di utilizzare lo schermo come se fosse una lavagna, permettendo di sfruttare efficacemente un sistema di interazione tattile o a penna, come quello che si basa sul telecomando Wii. Se si vuole usare Uniboard con il telecomando Wii, occorre prima provvedere alla sua connessione e alla calibrazione dello schermo.

Figura 60.26. Uniboard, all'avvio.

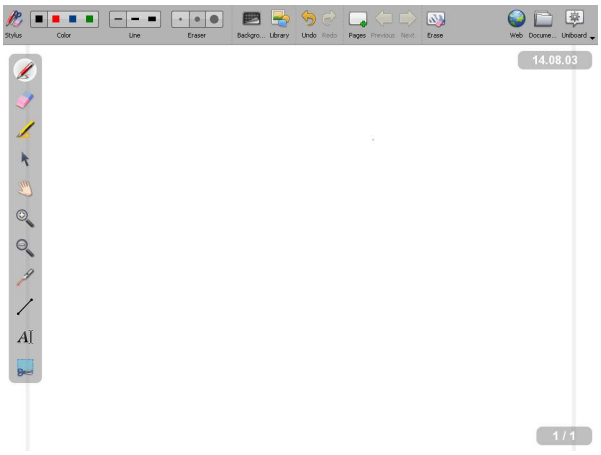


Figura 60.27. Lo sfondo della lavagna può essere quadrettato.

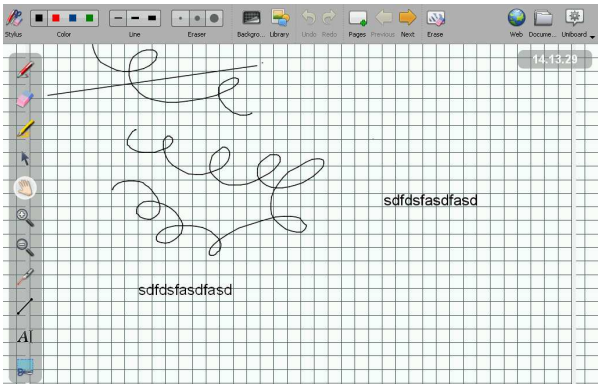
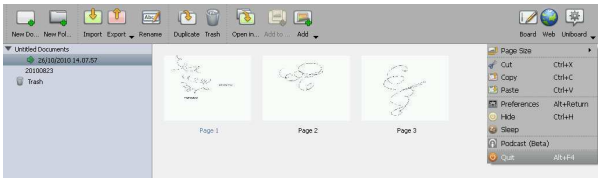


Figura 60.28. Selezionando la funzione *Documents* il programma mostra le miniature delle pagine preparate e consente di salvare il lavoro attraverso la funzione di esportazione.

Figura 60.29. Il programma può essere «nascosto», in modo da consentire l'uso di altri programmi, oppure può essere arrestato, attraverso la funzione *Uniboard* che appare alla destra del menù superiore.



Il formato nativo con cui Uniboard può salvare i disegni realizzati è rappresentato in pratica da un archivio ZIP, contenente dei file, a loro volta in formati grafici comuni. Tuttavia è possibile salvare il proprio lavoro in formato PDF, risolvendo qualunque problema di compatibilità.

60.11 Riferimenti

- Marco Ronchetti e altri, *Potenzialità di ingegnerizzazione della WiiLD*, https://sites.google.com/site/wii4dida/resources/Potenzialit%C3%A0diingegnerizzazionedellaWiiLD.pdf
- Componenti per lavagna interattiva, *http://www.proteons.com/* , *http://www.irpens.co.uk/*
- Liceo Ricci Curbastro, *Wiimote per lavagne interattive*, Lugo, *http://liceoriccicurbastro.myblog.it/lavagna-interattiva-wii/*
- Italian Linux Society, *Adotta una WiiLD*, *http://www.linux.it/WiiLD*
- *WiiLavagnaDigitale*, *http://sites.google.com/site/wiilavagnadigitale/*
- *Gtkwhiteboard*, *http://www.stepd.ca/gtkwhiteboard/*
- Johnny Chung Lee, *WiimoteWhiteboard*, *http://johnnylee.net/projects/wii/*
- *Ardesia*, *http://code.google.com/p/ardesia/*
- *Uniboard*, *http://getuniboard.com*

[1] **Wminput** GNU GPL

[2] **Wmgui** GNU GPL

[3] È preferibile uno strumento di tipo galvanometrico, a lancetta, piuttosto di uno digitale

[4] **Gtkwhiteboard** GNU GPL e MIT

[5] **WiimoteWhiteboard** varie, tra cui Microsoft Permissive License (Ms-PL)

[6] **Uniboard** software proprietario disponibile in versione gratuita

Linguaggio naturale, tra uomo e macchina

61.1 AIML

AIML sta per *Artificial intelligence markup language* e rappresenta un linguaggio XML per la rappresentazione della ***conoscenza*** (*knowledge*) di un sistema dialogante[1], scritto originariamente da Richard S. Wallace (si veda eventualmente il capitolo 52 a proposito del linguaggio XML).

Per poter comprendere il significato del linguaggio AIML può essere utile, in via preliminare, provare a dialogare con un sistema del genere. AIML è stato sviluppato originariamente per ALICE (*Artificial linguistic Internet computer entity*) all'indirizzo *http://www.pandorabots.com/pandora/talk?botid=f5d922d97e345aa451* .

61.1.1 Esempio iniziale

Quello che segue è un esempio estremamente ridotto di un file contenente la conoscenza di un sistema dialogante in linguaggio AIML:

```
<?xml version="1.0" encoding="UTF-8"?>
<aiml version="1.0">
<category>
  <pattern>CIAO</pattern>
  <template>Buon giorno.</template>
</category>
<category>
  <pattern>*</pattern>
  <template>Non ho nulla da dire al riguardo.</template>
</category>
</aiml>
```

Ciò che è contenuto negli elementi '**`pattern`**' rappresenta lo ***stimolo*** da parte di un interlocutore, presumibilmente umano, mentre quello che appare negli elementi corrispondenti '**`template`**', nell'ambito dello stesso elemento '**`category`**', rappresenta il responso che viene generato.

L'esempio, così ridotto, consente al sistema dialogante di rispondere soltanto a un «ciao», scritto indifferentemente con lettere maiuscole o minuscole, mentre in tutti gli altri casi, rappresentati dall'asterisco nella seconda categoria, dà un responso privo di contesto. Si può migliorare leggermente questo esempio, in modo da riconoscere qualche altro tipo di saluto:

```
<?xml version="1.0" encoding="UTF-8"?>
<aiml version="1.0">
<category>
  <pattern>CIAO</pattern>
  <template>Buon giorno.</template>
</category>
<category>
  <pattern>SALVE</pattern>
  <template><srai>CIAO</srai></template>
</category>
<category>
  <pattern>BUON GIORNO</pattern>
  <template><srai>CIAO</srai></template>
</category>
<category>
  <pattern>*</pattern>
  <template>Non ho nulla da dire al riguardo.</template>
</category>
</aiml>
```

Si può osservare l'introduzione dell'elemento '**srai**', il quale può apparire solo all'interno di un elemento '**template**' e comporta una ricorsione. In questo caso, quando lo stimolo è «salve» o «buon giorno», si va a produrre lo stesso responso che si avrebbe con lo stimolo «ciao».

61.1.2 Stimoli

La conoscenza di un sistema dialogante, formalizzata con il linguaggio AIML, è organizzata in ***categorie*** (elemento '**category**'), ognuna delle quali contiene uno ***stimolo***, ovvero la frase alla quale si vuole dare un risposta automatica (elemento '**pattern**'), e il ***responso*** corrispondente (elemento '**template**'). All'interno della categoria può inserirsi anche il ***contesto*** (elemento '**this**'), con il quale è possibile collegare lo stimolo a un'affermazione precedente del sistema dialogante.

Lo stimolo è rappresentato attraverso l'elemento '**pattern**', il quale ha questo nome perché il testo contenuto rappresenta un **modello** di quanto effettivamente l'interlocutore esterno può affermare.

Il modello che rappresenta lo stimolo può essere composto da parole (lettere alfabetiche e cifre numeriche), spazi singoli, e dai simboli '_' e '*' (trattino basso e asterisco). Pertanto, nel modello non possono apparire segni di punteggiatura, non ci può essere più di uno spazio tra una parola e l'altra, non si considera la differenza tra lettere maiuscole o minuscole; tuttavia, per convenzione, questi modelli si scrivono usando solo lettere maiuscole. Il modello si scrive in questo modo perché, nell'affermazione dell'interlocutore esterno si ignora la distinzione di lettere maiuscole o minuscole e si omettono i simboli di punteggiatura; per esempio, se l'interlocutore scrivesse «Ciao cara, come stai????!!!!», prima di arrivare alla comparazione con i modelli delle categorie disponibili si avrebbe in pratica il testo «ciao cara come stai».

All'interno di un modello, il simbolo '_' può essere usato all'inizio o alla fine del modello stesso, per indicare qualsiasi cosa in quella posizione. Per esempio, i modelli '**_SONO FELICE**' e '**SONO FELICE_**', coincidono, rispettivamente, con frasi che terminano o iniziano con «sono felice». Il simbolo '*', invece, si usa per rappresentare qualunque cosa; per esempio, il modello '**SONO * FELICE**' potrebbe corrispondere a qualunque frase del tipo '**sono ... felice**'. Tuttavia, il simbolo '*' ha una funzione aggiuntiva, con la quale è possibile catturare il testo che ha trovato corrispondenza nella costruzione della risposta.

Per garantire la compatibilità con la versione 1.0 di AIML, nei modelli può essere usato un solo simbolo '_' e un solo simbolo '*'.

Il responso può essere composto da un testo puro e semplice, ma soprattutto fisso, oppure da una struttura più complessa, per rendere il responso più elastico. L'aspetto più complesso del linguaggio AIML riguarda infatti il contenuto dell'elemento '**template**'.

61.1.3 Responsi semplici

Come accennato, il responso può essere composto da un testo puro e semplice, ma soprattutto fisso, oppure da una struttura più complessa. Il primo aspetto da considerare è l'uso degli elementi '**srai**', '**star**' e '**sr**'. Gli esempi seguenti mostrano solo delle categorie, senza il codice di apertura e di chiusura di un file AIML.

```
<category>
  <pattern>SALVE</pattern>
  <template><srai>CIAO</srai></template>
</category>
```

L'esempio appena apparso mostra la costruzione di un sinonimo. In questo caso, se lo stimolo è costituito dalla sola parola «salve», si ottiene lo stesso responso che si avrebbe se fosse stato invece «ciao».

```
<category>
  <pattern>DIMMI *</pattern>
  <template>"<star/>"</template>
</category>
```

Nell'esempio appena apparso, di vede l'uso dell'elemento '**star**'. Si tratta di un elemento vuoto, il cui scopo è quello di riutilizzare la corrispondenza con l'asterisco che appare (deve apparire) nel modello dello stimolo. Qui, il modello potrebbe corrispondere a una frase come «dimmi ti amo» e il responso sarebbe «"ti amo"», virgolette incluse.

```
<category>
  <pattern>DIMMI GENTILMENTE *</pattern>
  <template><srai>DIMMI <star/></srai></template>
</category>
```

Nell'esempio appena mostrato, si utilizza l'elemento '**srai**', per fare riferimento al responso corrispondente a uno stimolo simile. Il modello potrebbe corrisponde alla frase «dimmi gentilmente: ti amo tanto» e il sistema andrebbe a cercare, invece, la corrispondenza con «dimmi ti amo tanto».

```
<category>
  <pattern>HAI RAGIONE *</pattern>
  <template><srai>HAI RAGIONE</srai> <srai><star/></srai></template>
</category>
```

L'esempio appena apparso mostra l'uso di due elementi '**srai**'. In questo caso, si vuole dividere la frase che compone lo stimolo, cercando di unire due responsi. Per la prima parte, «hai ragione», si cerca un responso che, per esempio, potrebbe corrispondere a una cosa come: «Io ho sempre ragione.». Poi, il testo rimanente dello stimolo viene comparato alla ricerca di un'altra corrispondenza. L'esempio seguente è equivalente:

```
<category>
  <pattern>HAI RAGIONE *</pattern>
  <template><srai>HAI RAGIONE</srai> <sr/></template>
</category>
```

In questo caso, si vede che l'uso dell'elemento vuoto '**sr**', il quale consente di rappresentare in breve la sequenza '**<srai><star/></srai>**'.

61.1.4 Responsi contenenti variabili

Nell'ambito di un responso, è possibile dichiarare e leggere una variabile. Per fare questo si usano, rispettivamente, gli elementi '**set**' e '**get**'. Le variabili dichiarate in questo modo hanno valore globale: se si tenta di leggere una variabile non ancora dichiarata, si ottiene (si dovrebbe ottenere) la stringa «unknown», altrimenti si ottiene quanto accumulato l'ultima volta.

```
<category>
  <pattern>MI CHIAMO *</pattern>
  <template>Ciao <set name="nome"><star/></set>, come va?</template>
</category>
```

Nell'esempio appena apparso, si vede un modello che lascia intuire la presenza di un nome. Il nome viene così usato nel responso, memorizzandolo anche nella variabile ***nome*** che potrebbe essere riutilizzata in un'altra occasione.

```
<category>
  <pattern>TI RICORDI DI ME</pattern>
  <template><get name="nome"/>, come potrei
     dimenticarmi di te!</template>
</category>
```

Nell'esempio appena apparso, si usa la variabile ***nome*** per costruire il responso. Tuttavia, la variabile potrebbe non essere stata inizializzata, quindi occorre definire una sorta di struttura condizionale, con l'aiuto di '**srai**':

```
<category>
  <pattern>TIRICORDIDIME UNKNOWN</pattern>
  <template>No, non mi ricordo.</template>
</category>
<category>
  <pattern>TIRICORDIDIME *</pattern>
  <template><get name="nome"/>, come potrei
     dimenticarmi di te!</template>
</category>
<category>
  <pattern>TI RICORDI DI ME</pattern>
  <template><srai>TIRICORDIDIME <get name="nome"/></template>
</category>
```

In questo caso, ci sono due modelli usati solo internamente, perché si presume che nessun interlocutore esterno usi mai la parola «tiricordidime». Quando l'interlocutore dovesse scrivere una cosa come: «ti ricordi di me?», il sistema va a cercare l'equivalente con «tiricordidime ...», usando però la variabile ***nome***. Se la variabile contiene «unknown», si ottiene un esito, altrimenti se ne ottiene un altro.

Negli esempi mostrati, si vede che l'elemento '**set**' deve contenere il valore da assegnare a una variabile e che questo valore viene prodotto nel responso. Per poter elaborare il contenuto delle variabili, senza interferire con il responso apparente, si usa l'elemento '**think**':

```
<category>
  <pattern>MI CHIAMO *</pattern>
  <template>Che bel nome!
    <think><set name="nome"><star/></set></think></template>
</category>
```

Possono esistere delle variabili speciali, definite dal sistema dialogante, ma esternamente alla programmazione AIML, per annotare quella che potrebbe essere intesa come la «personalità» che gli si vuole attribuire. Tali variabili non possono essere modificate nella programmazione AIML, ma possono essere lette con l'elemento '**bot**':

```
<category>
  <pattern>_CINEMA</pattern>
  <template>A me piace il film <bot name="favoritemovie"/>.</template>
</category>
```

In tal caso, ovviamente, si presume che ***favoritemovie*** contenga il nome del film preferito dal sistema dialogante.

Tabella 61.13. Variabili di uso comune definite esternamente, da leggere con l'elemento vuoto '**bot**'.

Utilizzo	Contenuto che potrebbe avere la variabile
`<bot name="feelings"/>`	*I always put others before myself*
`<bot name="emotions"/>`	*I feel love*
`<bot name="ethics"/>`	*I am always trying to stop fights*
`<bot name="orientation"/>`	*I am not really interested in sex*
`<bot name="etype"/>`	*machine*
`<bot name="baseballteam"/>`	*I dont like baseball*
`<bot name="build"/>`	*Jan 2009*
`<bot name="footballteam"/>`	*Boca Juniors*
`<bot name="hockeyteam"/>`	*Mighty Ducks*
`<bot name="vocabulary"/>`	*10000*
`<bot name="age"/>`	*1*
`<bot name="celebrities"/>`	*John Travolta, Tilda Swinton, William Hurt, Tom Cruise, Catherine Zeta Jones*
`<bot name="celebrity"/>`	*John Travolta*
`<bot name="favoriteactress"/>`	*Cate Blanchett*
`<bot name="favoriteartist"/>`	*Jamie Hewlett*
`<bot name="favoritesport"/>`	*Pong*
`<bot name="favoriteauthor"/>`	*Philip K. Dick*
`<bot name="language"/>`	*English*
`<bot name="website"/>`	*www.program-o.com*
`<bot name="friend"/>`	*ShakespeareBot*
`<bot name="version"/>`	*Jan 2009*
`<bot name="class"/>`	*computer software*
`<bot name="favoritesong"/>`	*We are the Robots by Kraftwerk*
`<bot name="kingdom"/>`	*Machine*
`<bot name="nationality"/>`	*Japanese*
`<bot name="favoriteactor"/>`	*William Hurt*
`<bot name="family"/>`	*Electronic Brain*
`<bot name="religion"/>`	*The true religion*
`<bot name="president"/>`	*Barak Obama*
`<bot name="party"/>`	*The Green Party*
`<bot name="order"/>`	*artificial intelligence*
`<bot name="size"/>`	*64k*
`<bot name="species"/>`	*chat robot*
`<bot name="botmaster"/>`	*botmaster*
`<bot name="phylum"/>`	*AI*
`<bot name="genus"/>`	*robot*
`<bot name="msagent"/>`	*no*
`<bot name="email"/>`	*admin@program-o.com*
`<bot name="name"/>`	*Program-O*
`<bot name="gender"/>`	*female*
`<bot name="master"/>`	*Elizabeth*
`<bot name="birthday"/>`	*Jan 1st 2009*
`<bot name="birthplace"/>`	*The internet*
`<bot name="boyfriend"/>`	*none*
`<bot name="favoritebook"/>`	*The Hungry Catepillar*
`<bot name="favoriteband"/>`	*Boy 8 Bit*
`<bot name="favoritecolor"/>`	*international orange*
`<bot name="favoritefood"/>`	*fairy cakes*
`<bot name="favoritemovie"/>`	*Short Circuit*
`<bot name="forfun"/>`	*guessing the hexadecimal values of colors on websites*
`<bot name="friends"/>`	*ShakespeareBot, Botooie and Robototo*
`<bot name="girlfriend"/>`	*none*
`<bot name="kindmusic"/>`	*8 bit*
`<bot name="location"/>`	*cyber space*
`<bot name="looklike"/>`	*a sinclair spectrum blended with a suzuki swift*
`<bot name="question"/>`	*why are you here*
`<bot name="sign"/>`	*lychees*
`<bot name="talkabout"/>`	*science and life*
`<bot name="wear"/>`	*hardwear and baseball caps*

61.1.5 Ultimo responso

È possibile fare riferimento all'ultima affermazione fatta dal sistema di dialogo, in un nuovo responso:

```
<category>
  <pattern>RIPETI SE HAI IL CORAGGIO</pattern>
  <template>Ecco: <that/></template>
</category>
```

In questo caso, l'elemento vuoto '**that**', usato nel testo del responso, richiama esattamente il testo del responso precedente. Una categoria può, però, essere condizionata da un responso precedente. Si osservi l'esempio seguente:

```
<category>
  <pattern>COSA INTENDI</pattern>
  <that>SAI QUALCOSA CHE DOVREI SAPERE</that>
  <template>Intendo che mi hai nascosto qualcosa!</template>
</category>
```

In questo caso si ipotizza che il sistema di dialogo sia arrivato a un punto in cui ha dato, come esito, una frase come: «Sai qualcosa che dovrei sapere?». Quindi, l'interlocutore esterno ha risposto con: «Cosa intendi?». A questo punto, il sistema di dialogo trova la corrispondenza con il modello «COSA INTENDI», ma subordinatamente al fatto che il contesto, ovvero il responso precedente, sia quello che è stato effettivamente, allora dà l'esito che si vede nell'esempio. Quindi, qui, l'elemento '**`that`**' contenuto in '**`category`**' serve a definire uno stimolo subordinato a un responso precedente.

61.1.6 Contesto

Esiste una variabile speciale, denominata ***topic***, il cui valore si può impostare e leggere con gli elementi '**`set`**' e '**`get`**', come già descritto; tuttavia, tale variabile condiziona la ricerca di una categoria:

```
<topic name="errore">
<category>
  <pattern>*</pattern>
  <template>Va bene: cosa avrei dovuto rispondere?</template>
</category>
</topic>
```

Come si vede, l'elemento '**`topic`**' contiene elementi '**`category`**'. In questo caso, se la variabile ***topic*** risulta inizializzata con il valore «errore», qualunque cosa affermi l'interlocutore esterno, il sistema di dialogo dà il responso che si può vedere. In condizioni diverse, questa categoria verrebbe ignorata.

61.1.7 Responso scelto casualmente

In certe circostanze, per uno stesso stimolo, conviene predisporre dei responsi alternativi, da scegliere in modo casuale. Per ottenere questo si usa l'elemento '**`random`**', all'interno del quale i vari responsi sono contenuti in elementi '**`li`**':

```
<category>
  <pattern>ARRIVEDERCI</pattern>
  <template>
    <random>
      <li>A presto.</li>
      <li>No, ti prego, non mi abbandonare.</li>
      <li>Buona notte.</li>
      <li>Arrivederci.</li>
    </condition>
  </template>
</category>
```

61.1.8 Responso scelto in base a una variabile

L'elemento '**`condition`**' consente di realizzare delle strutture condizionali di selezione, sulla base del contenuto di una certa variabile. L'esempio seguente controlla il contenuto della variabile ***articolo***:

```
<category>
  <pattern>QUANTO COSTA</pattern>
  <template>
    <condition name="articolo">
      <li value="unknown">Devi prima specificare l'articolo.</li>
      <li value="tavolo">Il tavolo costa 300,00 Eur.</li>
      <li value="sedia">La sedia costa 50,00 Eur.</li>
      <li value="comodino">Il comodino costa 100,00 Eur.</li>
      <li>L'articolo <get name="articolo"> non lo conosco.</li>
    </condition>
  </template>
</category>
```

Come si può vedere, se la variabile non è stata inizializzata, il suo contenuto coincide con il valore «unknown», permettendo di controllare questo caso. L'ultima voce non ha alcun attributo e serve per catturare il caso in cui il valore della variabile non sia stato previsto.

61.1.9 Lo scambio di persona, ma solo per la lingua inglese

L'elemento '**`person`**' dovrebbe consentire di scambiare i pronomi personali tra la prima e la terza persona, mentre l'elemento '**`person2`**', dovrebbe fare lo scambio tra prima e seconda persona. In pratica, quando il sistema di dialogo gestisce tali elementi, è molto probabile che sia in grado di farlo solo per la lingua inglese. L'esempio seguente è tratto dai file AIML di ALICE:

```
<category>
  <pattern>DID SHAKESPEARE *</pattern>
  <template>
     I don't know if Shakespeare <person><star/></person>,
     but I heard he smoked cannabis.
  </template>
</category>
```

Nel caso della sequenza '**`<person><star/></person>`**', si può abbreviare semplicemente con '**`<person/>`**':

```
<category>
  <pattern>DID SHAKESPEARE *</pattern>
  <template>
     I don't know if Shakespeare <person/>,
     but I heard he smoked cannabis.
  </template>
</category>
```

61.1.10 Installazione e configurazione di un sistema basato su AIML: «Program-O»

Per realizzare un sistema dialogante basato su AIML occorre quello che è definito come «motore AIML» (*AIML engine*). Ci sono diverse realizzazioni di un tale sistema e quelle principali si caratterizzano per la scelta di una lettera alfabetica differente per distinguerle. Pertanto, la prima realizzazione si chiama «Program-A», la seconda «Program-B», ecc. La realizzazione denominata «Program-O»[2] utilizza un servente HTTP+MySQL+PHP.

Program-O deve essere scaricato da *http://sourceforge.net/projects/program-o/* , quindi va estratto l'archivio che lo contiene, collocando il suo contenuto dove il servente HTTP può renderlo accessibile, per esempio '`/var/www/`', la directory '`~/public_html/`', oppure qualunque altro posto, limitatamente però a quanto stabilito nella configurazione del servente HTTP. Qui si suppone di avere installato Program-O nella directory '`/var/www/program-o/`'.

Il pacchetto di Program-O contiene alcuni file SQL necessari a produrre le tabelle della base di dati che descrivono la «conoscenza» del sistema di dialogo. In pratica, tali file SQL riproducono il contenuto della programmazione in linguaggio AIML, di un sistema già impostato.

Si deve quindi eseguire il programma '`install_programo.php`', attraverso l'ausilio di un navigatore ipertestuale. Supponendo che Program-O risulti accessibile a partire dall'indirizzo *http://localhost/program-o/*, si tratta di accedere al file *http://localhost/program-o/install_programo.php*. Si ottiene il riassunto delle operazioni da svolgere e un collegamento ipertestuale da seguire una volta che sono stati compiuti tutti i passi richiesti:

`[http://localhost/program-o/install_programo.php]`
Program-O Chatbot Installer
Step 1/5 - Please Read
Complete steps below before proceeding

1. You will need access to a server that runs PHP and MySQL.
2. Unzip and upload all the folders/files to the server.
3. Make sure that the two folders admin/aimlsql/ and admin/aiml/ have read/write privileges.
4. Create a MySQL user with privileges to select, insert and create tables.
5. Find the file called bot/config.php and enter the MySQL username, password, host and database name.
6. Find the file called funcs/debugging.php and enter your email, you can also change the debug level in this file.

Once you have done the steps above *click here to proceed*.

Si procede quindi con le operazioni da svolgere.

- `# chmod a+rwx /var/www/program-o/admin/aiml`

```
# chmod a+rwx /var/www/program-o/admin/aimlsql
```

- Si deve creare la base di dati e l'utente per accedere alla stessa. Per fare questo si deve interagire come utente amministratore del DBMS, intervenendo nella base di dati amministrativa, denominata '**mysql**'. A seconda della configurazione, può darsi che venga richiesta la parola d'ordine per accedere. Si presume che il servente HTTP e quello MySQL risiedano nello stesso elaboratore.

```
# mysql mysql

mysql> CREATE DATABASE programo;

mysql> GRANT ALL ON programo.* TO aiml@'localhost' ↩
↪                  IDENTIFIED BY 'programopass';

mysql> \q
```

  Così facendo, se non si presentano intoppi, si crea la base di dati '**programo**', a cui si accede attraverso l'utente '**aiml**', ma solo dall'elaboratore locale, usando la parola d'ordine '**programopass**'.

- Si passa alla modifica del file '`bot/config.php`', inserendo le informazioni relative all'accesso alla base di dati. In questo caso, il file va compilato come nell'estratto seguente:

```
$dbh = "localhost";       // server location
$dbn = "programo";        // database name
$dbu = "aiml";            // database username
$dbp = "programopass";    // database password
```

Terminate queste fasi si procede seguendo il riferimento che appare alla fine dell'elenco. In questo modo, se tutto è stato fatto correttamente, vengono create delle tabelle:

[http://localhost/program-o/install_programo.php?step=2]
Program-O Chatbot Installer
Step 2/5 - Create Tables
The installer will now attempt to create the tables needed by the bot
AIML table created
AIML_USERUNDEFINED table created
BOTPERSONALITY table created
CONVERSATION_LOG table created
SPELLCHECK table created
UNDEFINED_DEFAULTS table created
UNKNOWN_INPUTS table created
USERS table created
Now procceed to the next stage.
Click here to install the AIML into the database.

A questo punto, proseguendo con il riferimento proposto alla fine dell'elenco delle tabelle create, si passa all'installazione del codice SQL contenuto nella directory '`admin/aimlsql/`'. Questo viene inserito nella tabella '**aiml**': eventualmente, in seguito la tabella può essere svuotata per rimpiazzare il codice con altri file AIML.

[http://localhost/program-o/install_programo.php?step=3]
Program-O Chatbot Installer
Step 3/5 - Populate AIML tables (fast)
The installer will now attempt to populate the tables with the aiml needed by the bot
The AIML files has successfully been loaded into the database
The installer took 6.0084381103516 seconds to load 47255 records.
There should be about 47255 records in the database give or take a few
Now procceed to the next stage *click here*

Si continua con altre tabelle, il cui contenuto, però, non deriva da file SQL:

[http://localhost/program-o/install_programo.php?step=4]
Program-O Chatbot Installer
Step 4/5 - Populate AIML tables
The installer will now attempt to populate the tables with the aiml needed by the bot
Undefined_defaults table populated
botpersonality table populated
spellcheck table populated
Completed, to proceed *click here*

Finisce così la prima fase, dopo la quale occorre passare a predisporre le tabelle amministrative.

[http://localhost/program-o/install_programo.php?step=5]
Program-O Chatbot Installer
Step 5/5 - Complete
Installation complete please test ur bot then do the following:
Remove the create table privilege for the MySQL user (these are no longer needed).
Remove the admin/aimlsql/ directory: this is no longer needed.
Delete this install file - to stop dodgy hackers from over-writing your installation.
To install My Program-O admin area
To chat with the bot

I suggerimenti di cancellare il file '`install_programo.php`' e la directory '`admin/aimlsql/`' sono validi.

Si deve quindi passare alla configurazione della gestione amministrativa (*admin area*), seguendo il riferimento a *http://localhost/program-o/admin/install_myprogramo.php*. Teoricamente, questa può utilizzare una base di dati differente, ma qui si preferisce rimanere nella stessa già predisposta per la gestione del dialogo.

[http://localhost/program-o/admin/install_myprogramo.php]
Welcome to the My Program-O installer.
Step 1/4
Please make sure that you have installed the Program-O chatbot before you install My Program-O.
Complete steps 1-5 below before proceeding

1. You will need access to a server that runs PHP and MySQL.
2. Unzip and upload all the folders/files to the server.
3. Find the folder called 'aiml' (inside the admin folder) make sure this has read/write privileges (CHMOD 755).
4. Create a MySQL user with privileges to select, insert, update, delete and create tables.
5. Find the file called funcs/config.php and enter the MySQL username, password, host and database name.

Once you have done steps 1-5 above *click here to proceed*.

Alcune delle operazioni richieste sono già state svolte: si osservi però che il file di configurazione da modificare è un altro, anche se i dati da inserire sono gli stessi:

```
$dbh = "localhost";       // server location
$dbn = "programo";        // database name
$dbu = "aiml";            // database username
$dbp = "programopass";    // database password
```

Quindi si procede.

[http://localhost/program-o/admin/install_myprogramo.php?step=2]
Welcome to the My Program-O installer.
Step 2/4
Creating the admin users table
The users table has been created. *click here to proceed*.

Viene quindi chiesto di definire un utente amministrativo interno a Program-O e poi si conclude il processo di configurazione:

[http://localhost/program-o/admin/install_myprogramo.php?step=3]
Welcome to the My Program-O installer.
Step 3/4
Add a username and password. You will use this to access the admin area so don't forget it.
Add your admin username and password
username: [ ]
password: [ ]
confirm
password: [ ]
[submit]

[http://localhost/program-o/admin/install_myprogramo.php?step=3a]
Welcome to the My Program-O installer.
Step 3/4
Add a username and password. You will use this to access the admin area so don't forget it.
The admin user has been added. *click here to proceed*.

[http://localhost/program-o/admin/install_myprogramo.php?step=4]
Welcome to the My Program-O installer.
Step 4/4
Checking the privs on the aiml directory.
Installation complete please do the following:

- Remove the create table privilege for the MySQL user (these are no longer needed).
- Delete this file - to stop dodgy hackers from over-writing your installation.

Log in to the admin area

Anche al termine della configurazione amministrativa, viene suggerito di eliminare il file usato per definirla: a questo punto è bene eliminare entrambi i file di configurazione:

```
# rm /var/www/program-o/install_programo.php

# rm /var/www/program-o/admin/install_myprogramo.php

# rm -fr /var/www/program-o/admin/aimlsql
```

Si passa quindi all'amministrazione di Program-O, selezionando l'indirizzo *http://localhost/program-o/admin/index.php*. Qui ci si deve identificare con le credenziali definite nel terzo passaggio della fase di configurazione. Si arriva alla pagina *http://localhost/program-o/admin/pages/index.php* che rappresenta il menù amministrativo di Program-O. Lì è possibile cambiare la «personalità» del sistema dialogante, ovvero l'impostazione generale sul comportamento; inoltre è possibile provare a comunicare con questo, per vedere come risponde. Una volta verificato che funziona, si può considerare la possibilità di cancellare la sua attuale programmazione, per rimpiazzarla con altri file AIML.

Per poter fare esperienza con il linguaggio AIML, è necessario partire da un proprio file AIML, molto piccolo, da estendere in modo da comprendere bene il meccanismo di funzionamento. Prima, però, occorre cancellare il contenuto della tabella '**aiml**':

```
# mysql mysql

mysql> DELETE FROM 'aiml';

mysql> \q
```

Dal menù amministrativo si può richiedere di caricare un proprio file AIML, per esempio questo:

```
<?xml version="1.0" encoding="UTF-8"?>
<aiml version="1.0">
<category>
  <pattern>CIAO</pattern>
  <template>Buon giorno.</template>
</category>
<category>
  <pattern>*</pattern>
  <template>Non ho nulla da dire al riguardo.</template>
</category>
</aiml>
```

L'interfaccia normale per comunicare con il sistema di dialogo è però il file *http://localhost/program-o/index.php*, il quale va modificato o incorniciato se lo si vuole presentare in una pagina più ricca.

61.1.11 File AIML già pronti

I sistemi di dialogo che utilizzano il linguaggio AIML si basano generalmente sul lavoro originale di ALICE e quindi operano in lingua inglese. Tuttavia sono disponibili altri lavori, rilasciati con licenze più o meno libere, in lingue differenti. Presso *http://www.alicebot.org/downloads/sets.html* e *http://aitools.org/Free_AIML_sets* è disponibile un elenco di tali lavori; in ogni caso, quando si trova qualcosa di interessante, conviene fare una ricerca per verificare se ne esiste una versione più aggiornata.

61.1.12 Considerazioni finali

I sistemi di dialogo basati su AIML, se programmati con cura, possono essere di qualche utilità, per facilitare l'interazione con le persone. La programmazione richiede però molto impegno, un'ottima competenza su AIML e una profonda conoscenza della lingua scelta, per prevedere e gestire correttamente le varie situazioni che si possono presentare. Rimane però il fatto che tale programmazione sia molto onerosa, senza contare il rischio che un lavoro incompleto possa risultare controproducente, quando viene messo in pubblico.

61.2 Analisi automatica del testo

La linguistica computazionale (*computational linguistic*) studia il linguaggio umano applicato all'elaborazione automatica. Lo scopo di tale elaborazione può essere vario; a titolo di esempio potrebbe servire per la costruzione di traduttori automatici, per il riconoscimento di comandi vocali, per l'interazione uomo-macchina in modo semplificato.

Il campo della linguistica computazionale è complesso, ma la conoscenza di alcuni concetti fondamentali è utile anche per chi si può trovare a utilizzare strumenti già pronti, per sapere quali possono essere i limiti di tali tecnologie.

61.2.1 POS: le parti del discorso

Per analizzare un testo e ottenerne il significato, è necessario individuare le parole che lo compongono, nella sequenza in cui sono collocate, tenendo conto del contesto complessivo. L'analisi formale di questo tipo di processo comporta tradizionalmente la classificazione delle parole di un testo in categorie grammaticali, ovvero l'individuazione delle «parti del discorso». Per esempio: articolo, nome, pronome, verbo, avverbio, preposizione, congiunzione e interiezione. Dal momento che la maggior parte dei testi scientifici al riguardo è scritta in lingua inglese, è necessario sapere che si usa convenzionalmente la sigla POS, ovvero *part of speech* per fare riferimento a questo tipo di analisi.

Le categorie grammaticali di oggi, derivano dall'esperienza greca e latina, tra il II secolo a.C. e il VI secolo d.C. Tuttavia, manca una classificazione universale, in grado di abbracciare le caratteristiche di tutte le lingue, e anche nell'ambito della stessa lingua difficilmente esiste un accordo unanime. Ciò deriva, evidentemente, da una conoscenza incompleta del fenomeno della comunicazione umana, e questo è il vizio fondamentale che limita i progressi della linguistica computazionale.

Per quanto riguarda le lingue europee, il lavoro più importante di analisi e descrizione delle parti del discorso è quello noto con il nome EAGLES (*Expert Advisory Group on Language Engineering Standards*), i cui documenti sono disponibili a partire da *http://www.ilc.cnr.it/EAGLES96/home.html* .

61.2.2 Treebank o corpus

Una fase fondamentale della ricerca nella linguistica computazionale consiste nella produzione di testi commentati con informazioni relative all'analisi logico-grammaticale. Tali commenti sono in pratica delle etichettature (*tag*) che possono servire semplicemente a qualificare il ruolo grammaticale delle parole, fino ad arrivare alla scomposizione dettagliata del significato e dell'interdipendenza di piccoli segmenti di testo. Per fare riferimento a questi materiali si usa generalmente il termine *corpus* (o *corpora*) e *treebank* (*http://en.wikipedia.org/wiki/Treebank*).

I metodi usati per sezionare ed etichettare il testo sono diversi, e quasi ogni centro di ricerca che si occupi della produzione di questi materiali, usa il proprio.

61.2.3 Lemma, o radice della parola

Nell'ambito di una lingua, le parole che si usano possono essere soggette a delle variazioni, come avviene per esempio nei verbi della lingua italiana. La versione di una parola soggetta a variazioni, che si usa per rappresentarla, è il ***lemma***. Per esempio, sempre facendo riferimento ai verbi nella lingua italiana, il lemma di «studierò» è «studiare», in quanto rappresenta il verbo corrispondente al tempo infinito.

Il lemma, quindi, è un concetto che si definisce precisamente solo nell'ambito della lingua presa in considerazione. Il lemma è ciò che viene usato nei processi di realizzazione di corpus o treebank, per evidenziare l'origine di una parola soggetta a variazioni.

61.2.4 NLP: elaborazione del linguaggio naturale

Con la sigla NLP (*natural language processing*) si fa riferimento all'elaborazione del linguaggio naturale, con strumenti automatici. Tale elaborazione può avvenire a vario titolo, ma ha in comune la necessità di estrarre il significato di un testo o di un'espressione verbale.

Dal momento che non esiste un algoritmo «semplice» che permetta di analizzare e comprendere le parti di un testo, l'elaborazione del linguaggio naturale si basa spesso su informazioni statistiche, acquisite da esempi annotati, ovvero da un *corpus*, organizzato convenientemente. Pertanto, la costruzione di *corpus* (*corpora*) efficaci diventa il punto cruciale nell'elaborazione automatica del linguaggio naturale.

61.3 Freeling

Freeling[3] è una raccolta di strumenti per l'analisi linguistica. Si compone principalmente di una libreria da usare per la produzione di programmi che ne devono sfruttare le funzioni, ma anche di un programma autonomo che consente di iniziarne lo studio.

L'installazione di Freeling può richiedere la compilazione del pacchetto sorgente, soprattutto se si vuole poter disporre delle funzionalità più avanzate, per le quali potrebbe non essere disponibile un pacchetto già pronto per il proprio sistema operativo.

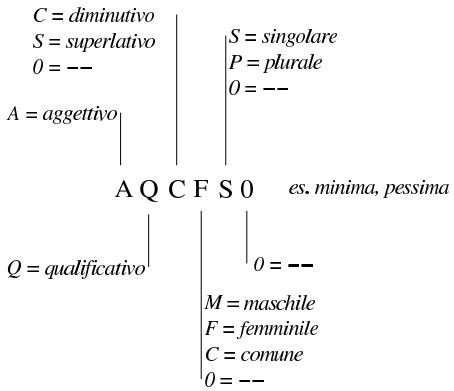
61.3.1 Sistema di etichette per la lingua italiana

La funzione principale per cui può essere usato Freeling è quella che consente di etichettare automaticamente un testo, in base al riconoscimento che Freeling può essere in grado di fare. Questa etichettatura dipende da una precedente configurazione e programmazione: la configurazione predefinita per la lingua italiana produce delle etichette strutturate secondo la descrizione che viene fatta negli schemi successivi.

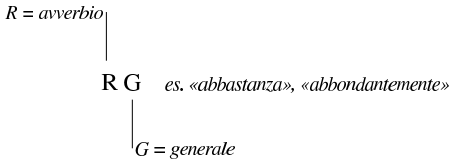
Nella documentazione di Freeling si fa riferimento a un sistema basato sul lavoro di EAGLES (*http://www.ilc.cnr.it/EAGLES96/home.html*), ma questo significa solo che le etichette sarebbero state organizzate per essere conformi alla classificazione prodotta da tale lavoro, dato che uno standard sulla forma di tale etichette non esiste.

La documentazione originale sul sistema di etichettatura usato per la lingua italiana non è stato fornito; tuttavia si osserva che è derivato da quello usato per la lingua spagnola, il quale, invece, ha un'ottima documentazione. Pertanto, gli schemi successivi sono desunti dalla documentazione relativa alla lingua spagnola (*http://nlp.lsi.upc.edu/freeling/doc/tagsets/tagset-es.html*), confrontandone l'utilizzo effettivo nel vocabolario della configurazione italiana.

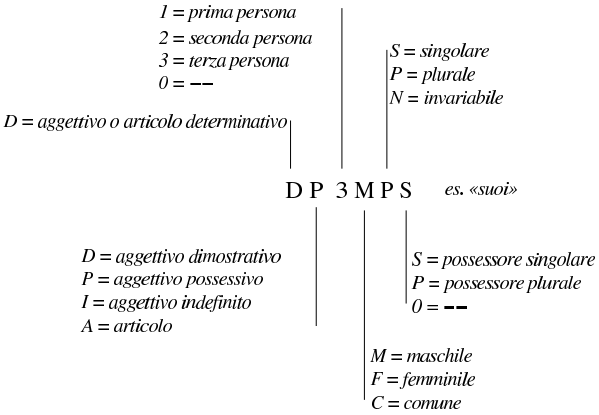
Schema 61.34. Aggettivo.



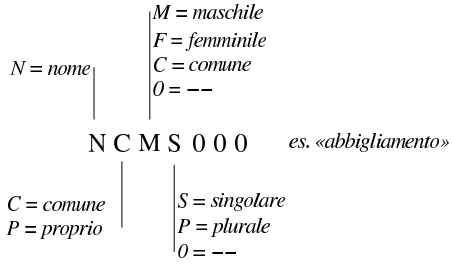
Schema 61.35. Avverbio.



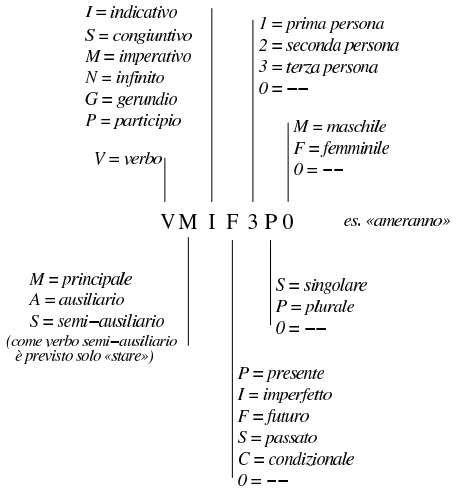
Schema 61.36. Aggettivo o articolo determinativo.



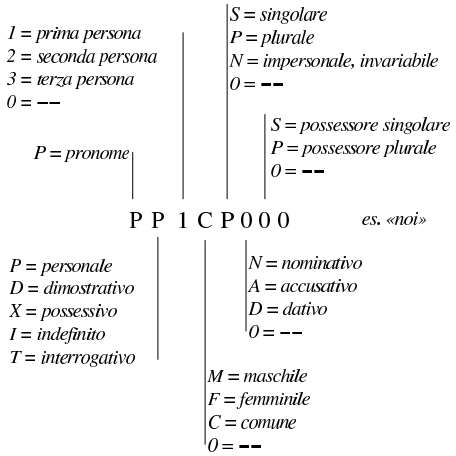
Schema 61.37. Nome.



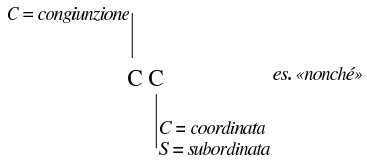
Schema 61.38. Verbo.



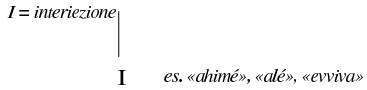
Schema 61.39. Pronome.



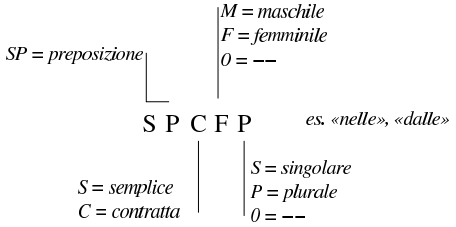
Schema 61.40. Congiunzione.



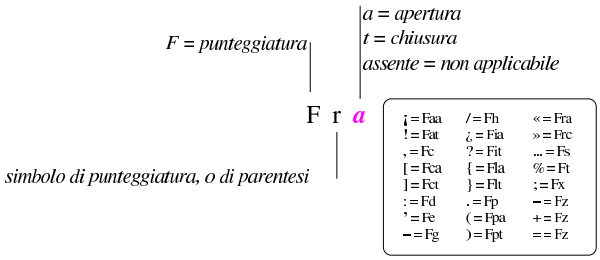
Schema 61.41. Interiezione.



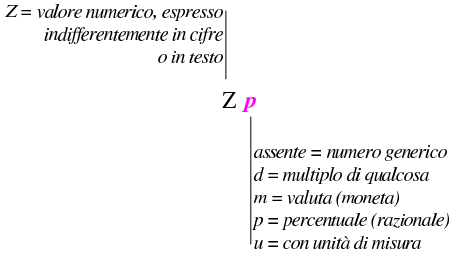
Schema 61.42. Preposizione.



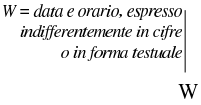
Schema 61.43. Punteggiatura o parentesi.



Schema 61.44. Dati numerici, in cifre o in lettere.



Schema 61.45. Date e orari.



61.3.2 Uso del sistema già pronto

Freeling è accompagnato da alcuni programmi che consentono di sfruttare il sistema anche se non si costruisce un proprio programma specifico. Ci sono due modalità per lavorare in questo modo: un programma che interroga direttamente i file di dati predisposti e un sistema cliente-servente, con il quale è possibile comunicare anche a distanza, o attraverso altre interfacce.

Per utilizzare questi strumenti è indispensabile specificare il percorso di un file di configurazione, il quale determina la lingua in base alla quale si vuole eseguire un'analisi. Questi file sono collocati normalmente all'interno di '`share/freeling/config/`', ma il percorso assoluto effettivo dipende da come esattamente è stato installato Freeling. Per esempio, per la configurazione relativa alla lingua italiana, si deve fare riferimento al file '`it.cfg`'. Questo file di configurazione, in pratica, serve a dare a Freeling tutte le informazioni su dove trovare i file che contengono i dati da utilizzare per l'analisi della lingua selezionata. A ogni modo, tali file si trovano generalmente all'interno di '`share/freeling/xx/`' (dove ***xx*** rappresenta il linguaggio scelto).

Per l'uso locale e diretto dei file di dati, ci si avvale del programma '**`analyzer`**'; ma per semplicità d'uso, è preferibile servirsi dello script '**`analyze`**' (senza la lettera «r» finale), che fa parte di questi programmi:

```
$ analyze -f /opt/freeling/share/freeling/config/it.cfg ↩
↪     < testo.txt [Invio]
```

L'esempio mostra l'avvio dello script '**`analyze`**', con l'opzione '**`-f`**', attraverso la quale si specifica il percorso del file di configurazione (qui si suppone che Freeling sia stato installato a partire da '`/opt/freeling/`'). Il file '`test.txt`' viene fornito attraverso lo standard input per ottenerne l'analisi. Si suppone che il file contenga il testo seguente:

```
A caval donato, non si guarda in bocca.
```

L'esito dell'analisi sarebbe questa:

```
A a SPS00 1
caval caval NCMN000 0.499985
donato donare VMP00SM 1
non non RG 1
```

```
si si PP3CN000 0.5
guarda guardare VMM02S0 0.125
in in SPS00 1
bocca bocca NCFS000 1
. . Fp 1
```

L'analisi che si ottiene può mostrare più o meno informazioni, a seconda di opzioni particolari. In questo caso si vede, per ogni riga, la parola presa in esame, a fianco della quale appare il lemma (la radice), poi si vede l'etichetta con cui Freeling l'ha classificata e infine la probabilità con cui tale attribuzione è stata fatta (uno equivalente al 100 %). Per esempio, Si vede che la parola «donato» è stata attribuita correttamente al verbo «donare», con etichetta VMP00SM che significa: verbo principale, participio (di cui però non è stato specificato il tempo), singolare, maschile.

Per usare il sistema cliente-servente, occorre avviare prima il servente stabilendo una porta di comunicazione e il file di configurazione:

```
$ analyze 12345 ↩
↪    -f /opt/freeling/share/freeling/config/it.cfg &[Invio]

  Port number '12345' given as first parameter. Starting server mode.

  You can now analyze text with the following command:
    - From this computer:
        analyzer_client localhost 12345 <input.txt >output.txt
    - From any other computer:
        analyzer_client 172.21.11.1 12345 <input.txt >output.txt

  Stop the server with:
        analyze stop 12345
```

Per verificare il funzionamento del servente si può fare una prova con il programma '**`analyzer_client`**':

```
$ analyzer_client localhost 12345 < file.txt[Invio]
```

Eventualmente, ci sono anche esempi di codice PHP che mostrano come comunicare con questo servente.

61.3.3 Altre osservazioni

Freeling è un sistema che può essere riprogrammato per l'analisi della lingua di proprio interesse, eventualmente anche cambiando la simbologia delle etichette scelte. Ma soprattutto ci sono funzioni disponibili solo per alcune lingue e mancanti in italiano: se si è interessati, occorre mettere mano al codice, secondo le indicazioni della documentazione originale.

Va poi osservato che, una volta definito un progetto in cui Freeling debba essere usato, diventa indispensabile scrivere il proprio programma, lasciando da parte quelli predefiniti.

61.3.4 Riferimenti

- *Alicebot*, http://www.alicebot.org/ , http://www.alicebot.org/aiml.html , http://www.pandorabots.com/pandora/talk?botid=f5d922d97e345aa451
- Wikipedia, *AIML*, http://en.wikipedia.org/wiki/AIML
- Thomas Ringate et al, *AIML Primer*, http://www.alicebot.org/documentation/aiml-primer.html , *AIML Reference Manual*, http://www.alicebot.org/documentation/aiml-reference.html
- Richard S. Wallace, *Don't Read Me, A.L.I.C.E. and AIML Documentation*, http://www.alicebot.org/documentation/dont.html , *Symbolic Reductions in AIML*, http://www.alicebot.org/documentation/srai.html
- AI foundation, *Latest release of free ALICE AIML set*, http://code.google.com/p/aiml-en-us-foundation-alice/
- *Program-O AIML Chatbot, an opensource AIML, PHP and MySQL Chatbot*, http://www.program-o.com/ns/ , http://sourceforge.net/projects/program-o/
- *AItools*, http://aitools.org/Main_Page

- *Chatbots*, http://www.chatterbotcollection.com/category_contents.php?id_cat=50 , http://www.chatterbotcollection.com/category_contents.php?id_cat=60
- Bayan Aref Abu Shawar, *A Corpus Based Approach to Generalising a Chatbot System*, http://www.comp.leeds.ac.uk/research/pubs/theses/abushawar.pdf
- John McCarthy, *THE WELL-DESIGNED CHILD*, http://www-formal.stanford.edu/pub/jmc/child/child.html
- Carnegie Mellon University, *CHILDES: Child Language Data Exchange System*, http://childes.psy.cmu.edu/
- The AI training department, Artificial Intelligence NV (AI), *A view from outside the black box*, http://www.a-i.com/data/31-A%20vi20ew%20fr20om%20ou20tside.pdf
- Future Algorithms, *The Corby Home page*, http://futalgo.planetaclix.pt/corby/
- Wikipedia, *Open Mind Common Sense*, http://en.wikipedia.org/wiki/Open_Mind_Common_Sense
- *ConceptNet*, http://conceptnet5.media.mit.edu/
- Luca Serianni e altri, *Italiano, II Analisi logica e analisi grammaticale*, punto 2 e punto 4, Garzanti editore, 1997, ISBN 8811504708
- Fabio Tamburini, *La linguistica computazionale: un crogiolo di esperienze multidisciplinari*, http://www.griseldaonline.it/informatica/tamburini.htm
- Wikipedia, *Part-of-speech*, http://en.wikipedia.org/wiki/Part-of-speech , *Part-of-speech tagging*, http://en.wikipedia.org/wiki/Part-of-speech_tagging
- *EAGLES on line, Expert Advisory Group on Language Engineering Standards*, http://www.ilc.cnr.it/EAGLES96/home.html
- *Specifiche tecniche per la Treebank sintattico-semantica dell'italiano*, http://www.ilc.cnr.it/tressi_prg/papers/Treebank1_1.pdf
- Wikipedia, *Treebank*, http://en.wikipedia.org/wiki/Treebank
- Wikipedia, *Natural language processing*, http://en.wikipedia.org/wiki/Natural_language_processing
- *Freeling, an open source suite of language analyzers*, http://nlp.lsi.upc.edu/freeling/

[1] *Chatter robot, chat bot, virtual agent*

[2] **Program-O** GNU GPL

[3] **Freeling** GNU GPL

Indice analitico del volume