

## Sistemi vari di composizione elettronica «

Introduzione a Lout .....	677
Collocazione dei componenti di Lout .....	677
Funzionamento .....	678
Esempio introduttivo .....	678
Concetti fondamentali di Lout .....	680
Caratteri speciali e stringhe letterali .....	681
Spazi e spaziature .....	682
Elementi essenziali di un documento Lout .....	682
Argomenti dei comandi e unità di misura .....	689
Rappresentazione simbolica della codifica .....	689
Caratteri da stampa .....	691
Display .....	692
Caratteristiche interne dei paragrafi .....	693
Testo letterale .....	695
Elenchi .....	695
Note .....	697
Figure e tabelle fluttuanti .....	698
Indici e riferimenti incrociati .....	700
Localizzazione .....	702
Personalizzazione dello stile .....	703
Riferimenti .....	704
Introduzione a LyX .....	705
Creazione di un documento .....	705
Struttura e stile .....	706
Modelli di documento .....	707
Automatismi .....	707
Riferimenti .....	707
Introduzione a HieroTeX .....	709
Installazione .....	709
Utilizzare HieroTeX .....	710
Codifica di HieroTeX .....	714
Riferimenti .....	720
Trasformazione in altri formati .....	721
DLH: trasforma LaTeX in HTML .....	721
Help2man: genera una pagina di manuale dalle informazioni fornite dal programma .....	722
Pstotext: estrae il testo da un file PostScript o PDF .....	723
Mswordview .....	723
Catdoc .....	723
Antiword .....	724
xLHTML .....	725

Collocazione dei componenti di Lout .....	677
Funzionamento .....	678
Esempio introduttivo .....	678
Concetti fondamentali di Lout .....	680
Caratteri speciali e stringhe letterali .....	681
Spazi e spaziature .....	682
Elementi essenziali di un documento Lout .....	682
Dichiarazione dello stile generale .....	682
Preambolo .....	683
Struttura del documento ordinario .....	684
Struttura della relazione tecnica .....	685
Struttura del libro .....	685
Struttura dei lucidi per lavagna luminosa .....	686
Sottostrutture .....	686
Suddivisione del testo .....	688
Argomenti dei comandi e unità di misura .....	689
Rappresentazione simbolica della codifica .....	689
Caratteri da stampa .....	691
Corpo .....	691
Display .....	692
Caratteristiche interne dei paragrafi .....	693
Interruzione e allineamento .....	693
Distanza tra le righe .....	693
Separazione in sillabe .....	694
Raccogliere tutto assieme .....	694
Testo letterale .....	695
Elenchi .....	695
Note .....	697
Note a piè pagina e note finali .....	697
Note a margine .....	697
Figure e tabelle fluttuanti .....	698
Figure .....	699
Tabelle .....	699
Indici e riferimenti incrociati .....	700
Riferimenti nel testo .....	700
Indice generale e indice analitico .....	701
Problemi connessi alla generazione dei riferimenti incrociati	702
Localizzazione .....	702
Configurazione di una localizzazione .....	703
Personalizzazione dello stile .....	703
Inclusione di altri stili .....	703
Veste grafica del documento .....	704
Particolarità del tipo di documento .....	704
Parte conclusiva .....	704
Riferimenti .....	704

Lout <sup>1</sup> è un sistema di editoria elettronica relativamente recente, che deriva dall'esperienza di \*roff e TeX, le cui potenzialità sono comparabili con quelle di TeX/LaTeX.

<http://lout.sourceforge.net>

## Collocazione dei componenti di Lout

Lout non è strutturato in una miriade di directory come succede alle distribuzioni LaTeX; è comunque necessario sapere dove sono state collocate alcune sue componenti. Per scoprirlo basta usare il comando seguente:

```
$ lout -v [Invio]
```

Con questo si potrebbe ottenere un messaggio simile a quello seguente:

```
Basser Lout Version 3.08 (May 1996)
Basser Lout written by:   Jeffrey H. Kingston (jeff@cs.usyd.edu.au)
Free source available from: ftp://ftp.cs.usyd.edu.au/jeff/lout
This executable compiled: 11:30:04 Aug 16 1998
System include directory: /usr/lib/lout/include
System database directory: /usr/lib/lout/data
Database index files created afresh automatically: yes
```

L'utente comune potrebbe non avere alcun bisogno di accedere a queste directory; comunque se si vuole realizzare un proprio stile personalizzato, occorre sapere che i file standard sono contenuti nella directory `'/usr/lib/lout/include/'` (in questo caso), essendo la directory delle inclusioni di sistema (come la definisce Lout).

Quando per qualche motivo si interviene nei file di configurazione di Lout contenuti in queste directory, è necessario sapere che poi Lout ha bisogno di generare dei file paralleli (per esempio da `'/usr/lib/lout/data/standard.ld'` viene generato `'/usr/lib/lout/data/standard.li'`). Lout fa le cose in modo automatico appena si accorge della necessità, tuttavia può darsi che in quel momento non abbia i permessi necessari per modificare o creare questi file. Bisogna tenere conto di questa possibilità, provvedendo a sistemare temporaneamente i permessi se ciò accade.

## Funzionamento

Allo stato attuale, Lout legge un sorgente e genera un risultato finale in PostScript.

Di solito si avvia l'eseguibile `'lout'` senza opzioni, con l'unico argomento costituito dal nome del file sorgente da convertire, in pratica secondo lo schema seguente:

```
lout file_sorgente > file_PostScript
```

Nonostante la ridirezione dello standard output, Lout emette altri messaggi attraverso lo standard error, meno dettagliati di quanto faccia TeX, ma altrettanto importanti. Una cosa da notare subito di Lout è che potrebbe essere necessario ripetere l'operazione di composizione più volte per poter risolvere i riferimenti incrociati, anche in presenza di documenti molto banali.

## Esempio introduttivo

La documentazione originale, scritta dallo stesso autore di Lout, parte da esempi molto semplificati per spiegare il comportamento di questo sistema di composizione; tuttavia, le possibilità del linguaggio di Lout potrebbero confondere; pertanto si preferisce mostrare qui un esempio un po' più complesso di quanto si veda di solito, ma allineato al genere di esempi già presentati per gli altri sistemi di composizione.

```
# Sorgente Lout di esempio. Per ottenere il risultato finale
# basta usare il comando:
# lout FILE_LOUT > FILE_PS

@SysInclude { doc }
@Document
  @InitialFont { Times Base 24p }
# @InitialBreak { adjust 1.2fx hyphen }
# @InitialSpace { lout }
# @InitialLanguage{ English }
# @PageHeaders { Simple }
# @FirstPageNumber { 1 }
# @ColumnNumber { 1 }
```

```
# @OptimizePages { No }
//
@Text @Begin

@Display @Heading { Introduzione a Lout }

Questo è un esempio di documento scritto con
Lout. Come si può vedere è stato definito
uno stile generale: doc.

@BeginSections
@Section
  @Title { Suddivisione del documento }
@Begin

@PP
Lo stile <doc> permette una suddivisione del
testo in sezioni, sottosezioni ed eventuali
sotto-sottosezioni
@End @Section

@Section
  @Title { Paragrafi }
@Begin

@PP
Il testo di un paragrafo inizia generalmente dopo il
simbolo "@PP", mentre non è presente la possibilità
di staccare i paragrafi solo attraverso una riga
vuota, come accade con TeX. Di solito, se non è
stato cambiato lo stile standard, la prima riga
appare rientrata.

@PP
Attenzione: gli spazi orizzontali,          vengono
rispettati!
@End @Section

@EndSections
@End @Text
```

È fondamentale per Lout che l'ultima riga utile del sorgente sia terminata da un'interruzione di riga. Se ci sono più righe vuote alla fine del sorgente, queste non creano problemi in ogni caso.

Supponendo di abbinare a questo file il nome `'esempio'`, si può utilizzare il comando seguente per comporre il documento e ottenere un file PostScript.

```
$ lout esempio > esempio.ps [Invio]
```

Per quanto strano possa sembrare, la prima volta vengono segnalati una serie di avvertimenti su dei riferimenti incrociati non risolti.

```
lout file "esempio":
25,1: unresolved cross reference @SectionList&&357.esempio.1
25,1: unresolved cross reference @SectionList&&357.esempio.1
35,1: unresolved cross reference @SectionList&&357.esempio.2
35,1: unresolved cross reference @SectionList&&357.esempio.2
```

Nel frattempo Lout ha creato alcuni file attorno a `'esempio'`: `'lout.li'` e `'esempio.ld'` (viene creato anche `'esempio.ps'`, ma non si tratta dell'edizione completa). La presenza di questi serve successivamente a risolvere parte o tutti i riferimenti incrociati.

```
$ lout esempio > esempio.ps [Invio]
```

In questo caso, la seconda volta che viene eseguito il comando si ottiene il risultato finale corretto.

Figura u93.4. Il risultato della composizione del sorgente Lout di esempio.

**Introduzione a Lout**

Questo è un esempio di documento scritto con Lout. Come si può vedere è stato definito uno stile generale: doc.

**1. Suddivisione del documento**

Lo stile «doc» permette una suddivisione del testo in sezioni, sottosezioni ed eventuali sotto-sottosezioni

**2. Paragrafi**

Il testo di un paragrafo inizia generalmente dopo il simbolo @PP, mentre non è presente la possibilità di staccare i paragrafi solo attraverso una riga vuota, come accade con TeX. Di solito, se non è stato cambiato lo stile standard, la prima riga appare rientrata.

Attenzione: gli spazi orizzontali, vengono rispettati!

Quando si modifica un documento dopo averlo già elaborato una volta con Lout, potrebbe essere opportuno eliminare i file generati in fase di composizione, in quanto questi possono produrre segnalazioni di errore fasulle, o comunque portare a un risultato finale errato.

Tra la documentazione che accompagna Lout si possono trovare i manuali di questo sistema di composizione, di solito anche in forma sorgente (sono scritti ovviamente in Lout). Questi possono essere ricompilati per ottenere un file PostScript e ciò permette di vedere cosa sia necessario fare di fronte a documenti più complessi. Per la precisione si tratta di documenti articolati in più sorgenti distinti, aggregati globalmente dal file ‘all’ (viene usato lo stesso nome per ogni manuale). Si intuisce che il comando di composizione debba essere simile a quello seguente (se non si dispone dei permessi di scrittura nella directory in cui si interviene, forse conviene lavorare su una copia), ma si può osservare che prima di riuscire a ottenere un risultato finale corretto, occorre riavviare il comando più volte, fino a quando non ci sono più riferimenti incrociati da risolvere:

```
§ lout all > risultato.ps [Invio]
```

## Concetti fondamentali di Lout

I comandi di Lout sono composti da **simboli**, ovvero delle parole chiave, che possono essere precedute e seguite da degli argomenti opzionali, il cui scopo è generalmente quello di intervenire su un **oggetto** posto alla loro destra (dopo le opzioni eventuali). È difficile esprimere il concetto in modo astratto, ma ancora più difficile è mostrarne un modello sintattico. All’interno di questi comandi vengono usate spesso le parentesi graffe, per raggruppare una serie di oggetti o una serie di argomenti; per questa ragione, nei modelli sintattici (semplificativi) che vengono mostrati, le parentesi graffe vanno intese in senso letterale, come facenti parte del comando.

Le parole chiave con cui sono definiti i simboli sono composte da **lettere**, che per Lout sono le lettere alfabetiche normali, maiuscole e minuscole (eventualmente anche accentate, ma in generale questo è meglio evitarlo), il carattere ‘@’ e il trattino basso (‘\_’). In generale, i simboli più comuni iniziano con il carattere ‘@’, in modo che la parola chiave che si ottiene non possa essere confusa con il testo normale, ma esistono comunque dei simboli che non rispettano questa consuetudine e di conseguenza vanno usati solo in contesti particolari. Il fatto che per Lout il carattere ‘@’ valga come una lettera normale, fa sì che possano esistere dei simboli (cioè delle parole chiave) che lo contengono all’interno; questo serve a capire che due parole chiave non possono essere aderenti, ma vanno spaziate in modo da consentire la loro individuazione.

Lout basa la sua filosofia su degli oggetti tipografici. Per comprenderlo si osservi l’esempio seguente.

```
Ecco qui: @I ciao a tutti. Sì, proprio @I { a tutti }.
```

Semplificando il concetto, un oggetto è una parola, compresa la punteggiatura che dovesse risultare attaccata a questa, oppure un raggruppamento di oggetti che si ottiene delimitandoli tra parentesi graffe. Osservando l’esempio, il simbolo ‘@I’ serve a ottenere il corsivo dell’oggetto che segue, dopo uno o più spazi che per i fini della composizione vengono ignorati. Pertanto, la prima volta che appare ‘@I’, questo serve a rendere in corsivo la parola ‘ciao’ che appare subito dopo, mentre la seconda, essendoci le parentesi graffe, il corsivo riguarda le parole ‘a tutti’. Si osservi che lo spazio contenuto tra le parentesi graffe, prima della parola ‘a’ e dopo la parola ‘tutti’, viene semplicemente ignorato ai fini della composizione tipografica. Si osservi ancora che il punto è stato lasciato fuori dal raggruppamento proprio per evitare che venga coinvolto dalla trasformazione in corsivo.

Da questo si può intendere che le parentesi graffe non servono solo a raggruppare degli oggetti, ma anche a dividere ciò che altrimenti sarebbe interpretato come un oggetto unico.

A fianco dell’uso delle parentesi graffe per delimitare un oggetto (raggruppando o dividendo degli oggetti preesistenti) si aggiungono le stringhe letterali, che sono a loro volta degli oggetti interpretati in modo letterale da Lout. Per esempio,

```
Il comando *@I ciao* genera il corsivo della parola «ciao».
```

si traduce in pratica nel testo seguente,

```
Il comando @I ciao genera il corsivo della parola «ciao».
```

dove si riesce a riportare nel risultato finale anche la lettera ‘@’, che altrimenti verrebbe assorbita per generare il corsivo.

Le stringhe vanno usate con parsimonia, perché generano degli oggetti che non possono essere suddivisi su più righe, comunque sono l’unico mezzo per rappresentare alcuni simboli che Lout altrimenti interpreterebbe.

Nell’esempio introduttivo si può notare l’uso del carattere ‘#’ che introduce un commento fino alla fine della riga. In pratica, questo serve a fare ignorare al sistema di composizione il testo che segue tale simbolo. Spesso, come è stato fatto nell’esempio, si commentano delle istruzioni di Lout che rappresentano un comportamento predefinito, per ricordare il punto in cui andrebbero collocate se fosse necessario cambiarne l’impostazione.

## Caratteri speciali e stringhe letterali

Fino a questo punto dovrebbe essere chiaro che le parentesi graffe, il carattere ‘@’, il carattere ‘#’ e gli apici doppi sono simboli che hanno un significato speciale, o possono essere interpretati in modo particolare. Oltre a questi se ne aggiungono altri e per tutti si pone il problema di poterli inserire nel testo in modo letterale, quando necessario. Ciò si ottiene con le stringhe letterali, delimitate tra apici doppi, come in parte è già stato notato. Usando le stringhe letterali resta comunque la difficoltà di rappresentare gli apici doppi, che così si ottengono con un carattere di escape aggiuntivo: la barra obliqua inversa. Questa, può essere usata **solo** all’interno delle stringhe letterali per mantenere invariato il significato letterale del carattere che la segue immediatamente; di conseguenza, per rappresentare una barra obliqua inversa, occorre usare una stringa letterale, confermando tale barra con un’altra barra obliqua inversa anteriore. La tabella u93.8 mostra l’elenco dei caratteri speciali per Lout e il modo di ottenerli all’interno delle stringhe letterali.

Tabella u93.8. Caratteri speciali di Lout e modo di ottenerli letteralmente all’interno delle stringhe.

Carattere speciale	Stringa letterale per ottenerlo	Carattere speciale	Stringa letterale per ottenerlo
(spazio)	" "	"	"\""
#	"#"	&	"&"

Carattere speciale	Stringa letterale per ottenerlo	Carattere speciale	Stringa letterale per ottenerlo
/	"/"	@	"@"
\	"\"	^	"^"
{	"{"		" "
}	"}"	~	"~"

## Spazi e spaziatore

Lout ha una gestione particolare degli spazi verticali e orizzontali. La prima cosa da notare è che le righe vuote non bastano a separare i paragrafi; per questo si usano comandi specifici, come '@PP' per esempio, che serve a introdurre il testo di un paragrafo. Pertanto, le righe vuote (una o più di una) vengono trattate al pari di spazi orizzontali aggiuntivi.

Gli spazi orizzontali normali, comprese le interruzioni di riga che si trasformano in spazi orizzontali, vengono rispettati; in particolare, il carattere di tabulazione viene interpretato come l'inserzione di otto spazi normali.

Questo comportamento predefinito di Lout potrebbe non essere desiderabile, per cui si può controllare attraverso l'opzione '@InitialSpace' che riguarda praticamente tutti i tipi di documento previsti da Lout. Il simbolo '@InitialSpace' prevede un argomento composto da una parola chiave (racchiusa tra parentesi graffe), che esprime il tipo di comportamento riferito alla gestione degli spazi:

Opzione	Descrizione
@InitialSpace { lout }	rappresenta l'impostazione predefinita, come è già stato descritto;
@InitialSpace { troff }	richiede un comportamento simile a quello di Troff;
@InitialSpace { tex }	richiede un comportamento simile a quello di TeX, in cui una sequenza di due o più spazi si traducono semplicemente in uno solo nel risultato finale.

## Elementi essenziali di un documento Lout

Lout, come LaTeX, è un po' delicato per quanto riguarda la sequenza di utilizzo di alcune istruzioni che definiscono la struttura del documento. A volte sono disponibili comandi differenti per fare le stesse cose, per esempio attraverso comandi abbreviati o semplificati. Benché si tratti di un sistema ben ordinato, si rischia di fare confusione. In questo senso, quando è possibile scegliere, qui vengono mostrate le forme più prolisse.

### Dichiarazione dello stile generale

Un sorgente Lout inizia generalmente con l'inclusione di un file esterno che serve a definire lo stile generale del documento. Nell'esempio introduttivo, dopo una serie di commenti, viene incluso lo stile 'doc', attraverso il simbolo '@SysInclude'. Il comando '@SysInclude { doc }' serve a inserire il contenuto del file 'doc' che si trova nella directory di inclusione nel sistema di Lout; in questo caso, seguendo quanto visto all'inizio del capitolo, si tratta di '/usr/lib/lout/include/'.

I tipi di documento principali che sono stati predisposti dall'autore di Lout sono:

Tipo	Descrizione
doc	un documento «ordinario» senza caratteristiche particolari;
report	il modello di una relazione tecnica;
book	un libro suddiviso in capitoli ed eventualmente in parti;
slides	un modello per le diapositive e per i lucidi da usare con la lavagna luminosa.

## Preambolo

Dopo l'inclusione dello stile si colloca normalmente un simbolo (di Lout) adatto al tipo di documento. Questo prevede una serie di opzioni e si conclude con due barre oblique. Nell'esempio introduttivo, trattandosi di un documento ordinario, si usava un preambolo simile a quello seguente; in questo caso però, vengono mostrate tutte le opzioni disponibili, indicate secondo il loro valore predefinito.

```
@SysInclude { doc }
@Document
@InitialFont { Times Base 12p }
@InitialBreak { adjust 1.2fx hyphen }
@InitialSpace { lout }
@InitialLanguage { English }
@PageHeaders { Simple }
@FirstPageNumber { 1 }
@ColumnNumber { 1 }
@OptimizePages { No }
//
```

Nell'esempio, dopo l'inclusione dello stile 'doc', appare il simbolo '@Document'; i simboli che si vedono sotto sono le sue opzioni e come tali vengono usati normalmente solo quando necessario per alterare alcune impostazioni predefinite. Può essere conveniente mettere tutte le opzioni disponibili, commentando quelle per le quali non c'è bisogno di alterarne l'impostazione predefinita, esattamente come si è fatto nell'esempio introduttivo. Ciò è utile quando si vuole rimaneggiare il documento senza fare troppa fatica, senza dover cercare le informazioni necessarie.

Il preambolo del documento di tipo 'report' è quello che si vede nell'esempio seguente:

```
@SysInclude { report }
@Report
@Title {}
@Author {}
@Institution {}
@DateLine { No }
@CoverSheet { Yes }
@InitialFont { Times Base 12p }
@InitialBreak { hyphen adjust 1.2fx }
@InitialSpace { lout }
@InitialLanguage { English }
@PageHeaders { Simple }
@ColumnNumber { 1 }
@FirstPageNumber { 1 }
@OptimizePages { No }
//
```

Si può osservare che alcuni simboli che descrivono delle opzioni hanno un argomento predefinito costituito da un oggetto nullo: '{}'. Nel seguito vengono mostrati i preamboli del documento di tipo 'book' e 'slide'.

```
@SysInclude { book }
@Book
@Title {}
@Author {}
@Edition {}
@Publisher {}
@BeforeTitlePage {}
@AfterTitlePage {}
@InitialFont { Times Base 12p }
@InitialBreak { adjust 1.2fx hyphen }
@InitialSpace { lout }
@InitialLanguage { English }
@PageHeaders { Titles }
@ColumnNumber { 1 }
@FirstPageNumber { 1 }
@IntroFirstPageNumber { 1 }
@OptimizePages { No }
//
```

```
@SysInclude { slides }
@OverheadTransparencies
@Title {}
@RunningTitle {}
@Author {}
@Institution {}
@DateLine { No }
@InitialFont { Times Base 20p }
@InitialBreak { ragged 1.2fx nohyphen }
@InitialSpace { lout }
@InitialLanguage { English }
@PageHeaders { Titles }
@FirstPageNumber { 1 }
@FirstOverheadNumber { 1 }
@FirstLectureNumber { 1 }
@OptimizePages { No }
//
```

Osservando gli esempi mostrati, si possono notare quali siano le opzioni più frequenti. Vale la pena di accennare subito ad alcune di queste.

Opzione	Descrizione
@Title	Come si può immaginare, il simbolo '@Title' serve come opzione per definire il titolo del documento; si può usare in tutte le situazioni in cui ciò possa avere senso. Infatti, nel caso del documento ordinario non è prevista questa possibilità.
@InitialFont	Il simbolo '@InitialFont' serve a definire il tipo di carattere e il corpo da utilizzare nel testo normale.
@InitialBreak	Il simbolo '@InitialBreak' serve a definire le caratteristiche dei paragrafi di testo normali; in particolare l'allineamento, la distanza tra le righe e l'attivazione o meno della separazione in sillabe delle parole.
@InitialSpace	Come già descritto in precedenza, il simbolo '@InitialSpace' serve a definire il comportamento di Lout nei confronti degli spazi, nel senso di stabilire se questi devono essere rispettati oppure se si deve fare come TeX che li ricompatta sempre.
@InitialLanguage	Il simbolo '@InitialLanguage' serve a adattare il comportamento di Lout in funzione del tipo di linguaggio. L'utilizzo di un linguaggio implica, per esempio, la scelta del modo in cui possono essere separate le sillabe e i nomi di alcune definizioni standard del documento.

#### Struttura del documento ordinario

Il contenuto di un documento scritto con Lout è racchiuso all'interno di uno o più ambienti specifici per il tipo di stile prescelto. Per esempio, nel caso del documento ordinario, si usano i comandi '@Text @Begin' e '@End @Text':

```
@SysInclude { doc }
@Document
  @InitialFont { Times Base 12p }
  ...
  //
  @Text @Begin
  ...
  @End @Text
```

Volendo, i due simboli possono essere posti anche su righe differenti, in modo da rendere più chiaro il loro significato, anche se questo è però contrario alla filosofia di Lout.

```
...
@Text
@Begin
...
@End
@Text
```

Il simbolo '@Text' iniziale ha come argomento il testo del documento; in teoria questo potrebbe essergli fornito attraverso le parentesi graffe:

```
...
@Text {
...
}
```

In pratica, questo modo di scrivere il sorgente Lout potrebbe essere troppo complicato; così, di fronte a oggetti di dimensioni molto grandi si preferisce utilizzare i delimitatori '@Begin' e '@End', nel modo mostrato.

In generale, un comando che può ricevere un oggetto delimitato dai simboli '@Begin' e '@End' può riceverlo anche se questo è racchiuso solo da parentesi graffe, mentre il contrario non è sempre possibile. In generale, si trova questa possibilità solo nei comandi che delimitano una struttura a larga scala.

L'ambiente '@Text' di questo tipo di documento può contenere anche delle sezioni e un'appendice, in modo simile a quello che viene mostrato nelle sezioni seguenti che fanno riferimento agli altri tipi di stile utilizzabile. In parte questo è già stato visto nello stesso esempio introduttivo.

#### Struttura della relazione tecnica

La relazione tecnica, ovvero lo stile 'report', prevede dopo il preambolo l'inserimento facoltativo dell'ambiente '@Abstract'; successivamente prevede la presenza di uno o più ambienti '@Section', infine è ammessa la presenza di uno o più ambienti '@Appendix'.

```
@SysInclude { report }
@Report
  @Title {}
  ...
  //
  @Abstract
  @Title {}
  ...
  @Begin
  ...
  @End @Abstract
  @Section
  @Title {}
  ...
  @Begin
  ...
  @End @Section
  @Appendix
  @Title {}
  ...
  @Begin
  ...
  @End @Appendix
  ...
```

Si può intuire il senso di questi ambienti e il ruolo dell'opzione '@Title' che appare in ognuno di questi: la relazione tecnica può avere un riassunto introduttivo, si suddivide in sezioni e può terminare con un'appendice. A loro volta, le sezioni e le appendici si possono scomporre, nel modo che viene mostrato in seguito.

#### Struttura del libro

Il libro, ovvero lo stile 'book', prevede dopo il preambolo l'inserimento facoltativo degli ambienti '@Preface' e '@Introduction'. Successivamente il documento viene suddiviso in capitoli, attraverso gli ambienti '@Chapter', e può concludersi con una serie di appendici.

```
@SysInclude { book }
@Book
  @Title {}
  ...
  //
  @Preface
  @Title { Prefazione }
  ...
  @Begin
  ...
  @End @Preface
  @Introduction
  @Title { Introduzione }
  ...
  @Begin
  ...
  @End @Introduction
  @Chapter
  @Title {}
  ...
  @Begin
  ...
  @End @Chapter
  ...
  @Appendix
  @Title {}
  ...
  @Begin
  ...
  @End @Appendix
  ...
```

Quello che si vede sopra è la struttura che potrebbe avere un documento di tipo 'book' che include sia l'ambiente '@Preface' che '@Introduction'.

I capitoli possono suddividersi ulteriormente in sezioni, nello stesso modo in cui si possono inserire le sezioni nell'ambiente '@Text' quando si usa lo stile 'doc'.

I capitoli potrebbero essere raggruppati in parti, ma non esistendo un ambiente del genere, si annota l'inizio di una nuova parte tra le opzioni del capitolo che si intende debba seguirla immediatamente. L'esempio seguente mostra il capitolo intitolato 'Primo approccio' che si trova a essere il primo della parte 'Principianti', indicata come 'Parte IV'.

```
@Chapter
  @PartNumber { Parte IV }
  @PartTitle { Principianti }
  @Title { Primo approccio }
  @Begin
  ...
  @End @Chapter
```

Perché la suddivisione in parti venga presa in considerazione, è necessario che l'opzione '@PartTitle' abbia un argomento non vuoto, cioè disponga di un titolo. Se inoltre si vuole inserire del testo tra il titolo della parte e l'inizio del capitolo, occorre utilizzare l'opzione '@PartText' che prende come argomento il testo in questione.

```
@Chapter
  @PartNumber { Parte IV }
  @PartTitle { Principianti }
  @PartText {
    ...
    ...
    ...
  }
  @Title { Primo approccio }
  @Begin
  ...
  @End @Chapter
```

#### Struttura dei lucidi per lavagna luminosa

Le diapositive, ovvero lo stile 'slides', prevede dopo il preambolo la suddivisione del documento in ambienti '@Overhead', che poi non possono contenere altre strutture a larga scala (in pratica non possono contenere sezioni o simili).

```
@SysInclude { slides }
@OverheadTransparencies
  @Title {}
  ...
  //
  @Overhead
  @Title {}
  ...
  @Begin
  ...
  @End Overhead
  ...
```

#### Sottostrutture

L'ambiente '@Text' di un documento ordinario e i capitoli di un libro possono contenere delle sezioni, delimitate dai simboli '@BeginSections' e '@EndSections'. Si osservino gli esempi seguenti, dove nel primo caso vengono inserite delle sezioni all'interno di un documento ordinario, mentre nel secondo all'interno di un capitolo di un libro.

```
@SysInclude { doc }
@Document
  ...
  //
  @Text @Begin
  ...
  @BeginSections
  @Section
  @Title {}
  ...
  @Begin
  ...
  @End @Section
  ...
```

```
@EndSections
...
@End @Text
```

```
@SysInclude { book }
@Book
  @Title {}
  ...
  //
  @Chapter
  @Title {}
  ...
  @Begin
  ...
  @BeginSections
  @Section
  @Title {}
  ...
  @Begin
  ...
  @End @Section
  ...
  @EndSections
  ...
  @End @Chapter
  ...
```

All'interno delle sezioni, comprese quelle delle relazioni tecniche, è ammissibile la suddivisione in sottosezioni delimitate dai simboli '@BeginSubSections' e '@EndSubSections'.

```
@Section
  @Title {}
  ...
  @Begin
  ...
  @BeginSubSections
  @SubSection
  @Title {}
  ...
  @Begin
  ...
  @End @SubSection
  ...
  @EndSubSections
  ...
  @End @Section
```

Nello stesso modo funzionano anche le sotto-sottosezioni, attraverso la delimitazione dei simboli '@BeginSubSubSections' e '@EndSubSubSections'.

```
@SubSection
  @Title {}
  ...
  @Begin
  ...
  @BeginSubSubSections
  @SubSubSection
  @Title {}
  ...
  @Begin
  ...
  @End @SubSubSection
  ...
  @EndSubSubSections
  ...
  @End @SubSection
```

Lout non prevede ulteriori suddivisioni; comunque, anche le appendici possono essere suddivise in modo simile in sottoappendici e sotto-sottoappendici.

```
@Appendix
  @Title {}
  ...
  @Begin
  ...
  @BeginSubAppendices
  @SubAppendix
  @Title {}
  ...
  @Begin
  ...
  @BeginSubSubAppendices
  @SubSubAppendix
  @Title {}
  ...
  @Begin
  ...
  @End @SubSubAppendix
  ...
  @EndSubSubAppendices
  ...
```

```
@End @SubAppendix
...
@EndSubAppendices
...
@End @Appendix
```

## Suddivisione del testo

Come già accennato in precedenza, Lout impone l'indicazione esplicita dell'inizio di un blocco di testo, ovvero un paragrafo. Gli spazi verticali non servono allo scopo come accade con LaTeX. In generale, si utilizzano i comandi '@PP' e '@LP'; il primo inizia un paragrafo normale, il secondo un paragrafo allineato a sinistra. La differenza sta nel fatto che normalmente '@PP' fa rientrare leggermente la prima riga, mentre il secondo no.

```
@PP
Lo stile «doc» permette una suddivisione del
testo in sezioni, sottosezioni ed eventuali
sotto-sottosezioni
```

L'esempio che si vede sopra è esattamente uguale, come risultato, a quello seguente:

```
@PP Lo stile «doc» permette una suddivisione del
testo in sezioni, sottosezioni ed eventuali
sotto-sottosezioni
```

La separazione del testo in paragrafi comporta normalmente l'inserzione di uno spazio verticale aggiuntivo tra la fine di uno e l'inizio del successivo. Per ottenere semplicemente l'interruzione di una riga (il ritorno a capo) si può utilizzare il comando '@LLP'.

```
@PP
Lo stile «doc» permette una suddivisione del
testo in sezioni, sottosezioni ed eventuali
sotto-sottosezioni;
@LLP
le sotto-sotto-sottosezioni non esistono.
```

L'esempio che si vede sopra è esattamente uguale, come risultato, a quello seguente:

```
@PP Lo stile «doc» permette una suddivisione del
testo in sezioni, sottosezioni ed eventuali
sotto-sottosezioni; @LLP le sotto-sotto-sottosezioni
non esistono.
```

Un paragrafo può essere messo in risalto (*display*) staccandolo dal resto del documento. Il comando '@DP' serve a ottenere un paragrafo senza il rientro della prima riga, un po' più staccato verticalmente da quello precedente.

```
@DP
Questo paragrafo risulta staccato meglio da quello
precedente.
```

Per aumentare questo distacco dal resto del testo, si possono usare più simboli '@DP' ripetutamente.

```
@DP
@DP
@DP
Questo paragrafo risulta molto staccato da quello
precedente.
```

Per richiedere espressamente il salto pagina in un punto del documento, si può usare il comando '@NP' il cui scopo è proprio quello di iniziare un paragrafo nuovo a partire dalla prossima colonna. Il paragrafo in questione non ha il rientro iniziale della prima riga.

```
@NP
Questo paragrafo inizia in una colonna, o in una pagina nuova.
```

Infine, il comando '@CNP' inizia un paragrafo che potrebbe essere spostato all'inizio della prossima colonna, o della prossima pagina, se non c'è abbastanza spazio per scrivere alcune righe.

È importante osservare che i simboli di questi comandi non prevedono argomenti e il testo del paragrafo che viene collocato dopo di questi non ne è legato in alcun modo. In pratica, il compito di '@PP' è quello di inserire uno spazio verticale aggiuntivo e memorizzare da qualche parte che il testo deve iniziare con una riga rientrata. Se si utilizza per due volte '@PP', si ottiene uno spazio di separazione verticale doppio.

## Argomenti dei comandi e unità di misura

Fino a questo punto sono stati mostrati molti comandi di Lout senza descriverne il significato. Alcuni di questi richiedono un argomento composto dall'unione di più informazioni, come nell'esempio seguente,

```
@InitialFont { Times Base 20p }
@InitialBreak { ragged 1.2fx nohyphen }
```

dove ognuno dei due simboli mostrati richiede l'indicazione di tre argomenti raggruppati attraverso l'uso delle parentesi graffe.

Gli argomenti di un simbolo di Lout possono essere richiesti prima o dopo il simbolo stesso. Quando si tratta di informazioni numeriche che rappresentano una dimensione, queste sono intese essere espresse secondo un'unità di misura predefinita, oppure secondo l'unità stabilita da una lettera indicata subito dopo il numero. Per esempio, '@20p' rappresenta 20 punti tipografici.

Tabella u93.36. Unità di misura principali di Lout.

Lettera	Unità di misura corrispondente
'c'	Centimetri.
'i'	Pollici ('1i' = '2.54c').
'p'	Punti tipografici ('72p' = '1i').
'm'	Quadrati ('12m' = '1i').
'f'	La dimensione attuale del corpo.
's'	La dimensione attuale di uno spazio (spaziatura).
'v'	La distanza attuale tra le righe.

Naturalmente, i valori che esprimono quantità non intere possono essere espressi utilizzando il punto di separazione tra la parte intera e quella decimale.

A volte, alcuni argomenti numerici devono essere conclusi con una lettera 'x' (dopo l'indicazione dell'unità di misura). Intuitivamente si può associare questo fatto all'idea che si tratti di un valore che debba essere moltiplicato a qualcosa per ottenere il risultato, ovvero che si tratti di un dato relativo. Per esempio, il valore '@1.2fx' del comando seguente rappresenta il 120% dell'attuale dimensione dei caratteri (il corpo del carattere moltiplicato per 1,2):

```
@InitialBreak { ragged 1.2fx nohyphen }
```

La ragione precisa non è questa, ma la spiegazione approssimativa data può almeno essere utile per accettare la cosa temporaneamente, finché non si intende affrontare lo studio approfondito di Lout.

## Rappresentazione simbolica della codifica

Come in tutti i sistemi di composizione tipografica, anche Lout ha un modo per rappresentare simbolicamente alcuni caratteri particolari. In precedenza si è accennato alla possibilità di inserire nel testo i caratteri speciali che Lout tende a interpretare in modo particolare, attraverso le stringhe letterali. Lout permette anche di usare dei simboli nella forma seguente:

```
@Char nome
```

Ciò permette di rappresentare qualunque carattere: sia l'alfabeto normale, sia i simboli di punteggiatura, sia qualunque altro simbolo speciale. Per esempio, '@Char A' è la lettera 'A' maiuscola, mentre '@Char a' è la lettera 'a' minuscola. La tabella u93.38 elenca alcuni dei comandi che possono essere utili per rappresentare le lettere accentate e altri caratteri importanti.

Tabella u93.38. Alcuni comandi per le lettere accentate di Lout.

@Char acute	á	@Char Aacute	Á
	â	@Char Aacute	Â
@Char acircumflex		@Char Acircumflex	Â
@Char agrave	à	@Char Agrave	À
@Char aring	å	@Char Aring	Å
@Char atilde	ã	@Char Atilde	Ã

@Char adieresis	ä	@Char Adieresis	Ä
@Char ae	æ	@Char AE	Æ
@Char ccedilla	ç	@Char Ccedilla	Ç
@Char eacute	é	@Char Eacute	É
	ê	@Char Ecircumflex	Ê
@Char ecircumflex		@Char Egrave	È
@Char egrave	è	@Char Edieresis	Ë
@Char edieresis	ë	@Char Iacute	Í
@Char iacute	í	@Char Icircumflex	Î
	î	@Char Igrave	Ì
@Char icircumflex		@Char Idieresis	Ï
@Char igrave	ì	@Char Ntilde	Ñ
@Char idieresis	ï	@Char Oacute	Ó
@Char ntilde	ñ		Ô
@Char oacute	ó	@Char Ocircumflex	Û
	ô	@Char Ograve	Û
@Char ocircumflex		@Char Oslash	Ø
@Char ograve	ò	@Char Otilde	Õ
@Char oslash	ø	@Char Odieresis	Ö
@Char otilde	õ	@Char germandbls	ß
@Char odieresis	ö	@Char uacute	Ú
@Char germandbls	ß		Û
@Char uacute	ú	@Char Ucircumflex	Û
	û	@Char Ugrave	Û
@Char ucircumflex		@Char Udieresis	Û
@Char ugrave	ù	@Char Yacute	Ý
@Char udieresis	ü	@Char Ydieresis	ÿ
@Char yacute	ý		
@Char ydieresis	ÿ		

Quando si vuole rappresentare in questo modo una lettera accentata o un altro carattere tipografico speciale, come parte di una parola, si è costretti a inserire il comando relativo all'interno di parentesi grafiche. Per comprendere il problema, si pensi alla possibilità di scrivere la parola «così» indicando la lettera 'i' accentata con il comando '@Char igrave'. L'esempio seguente è errato:

```
cos@Char igrave # errato
```

Infatti, Lout non è in grado di riconoscere il simbolo '@Char', dal momento che questo risulta attaccato ad altre lettere (e bisogna ricordare che per Lout il carattere '@' è una lettera come le altre). Il modo giusto di scrivere quella parola è quindi:

```
cos{ @Char igrave }
```

Oltre alla codifica normale, Lout mette a disposizione anche un alfabeto simbolico attraverso l'uso di comandi '@Sym'.

```
@Sym nome
```

Per conoscere i nomi che si possono utilizzare per ottenere le lettere greche e altri caratteri simbolici, deve essere letta la documentazione originale.

## Caratteri da stampa

La scelta del carattere da stampa avviene prevalentemente attraverso una serie di comandi che riguardano la forma del carattere riferita allo stile attuale o l'indicazione precisa dello stile (ovvero della famiglia) e della forma.

```
@B { testo_in_neretto }
```

```
@I { testo_in_corsivo }
```

```
@BI { testo_in_neretto_corsivo }
```

```
@R { testo_in_tondo }
```

```
@S { testo_in_maiuscolo }
```

Quelli che si vedono sono i comandi per ottenere una variazione della forma all'intero dello stile attuale del testo circostante. A questi comandi si affianca anche il comando per ottenere uno stile dattilografico, che pur non essendo semplicemente una variazione di forma, data la sua importanza nei documenti a carattere tecnico lo si abbina idealmente a questi per semplicità:

```
@F { testo_in_dattilografico }
```

Per cambiare in modo esplicito lo stile del carattere si può usare il comando '@Font' che richiede l'indicazione del nome dello stile, della forma e ovviamente del testo su cui intervenire:

```
{ stile forma } @Font { testo }
```

Gli stili più comuni sono: 'Times', 'Helvetica' e 'Courier'. A questi si aggiungono anche delle specie simboliche, come 'symbol', solo che a questa si accede generalmente attraverso il comando '@Sym'.

La forma viene specificata attraverso una parola chiave che può essere: 'Base', per indicare un carattere tondo chiaro; 'slope', per indicare una forma corsiva o inclinata (a seconda della disponibilità di quel tipo di stile); 'Bold', per indicare il neretto; 'BoldSlope', per indicare un neretto-corsivo. Naturalmente, la forma richiesta è ottenuta solo se lo stile scelto lo permette.

```
Lo stile { Courier BoldSlope } @Font { doc } permette una suddivisione del testo in sezioni, sottosezioni ed eventuali sotto-sottosezioni.
```

L'esempio che si vede sopra, serve a fare in modo che la parola 'doc' sia resa con lo stile 'Courier' in neretto-inclinato. Un risultato simile può essere ottenuto attraverso il comando '@F', nel modo seguente:

```
Lo stile @F { @BI { doc } } permette una suddivisione del testo in sezioni, sottosezioni ed eventuali sotto-sottosezioni.
```

I comandi di Lout per la definizione della forma non sono cumulativi, ed è per questo che esiste il comando '@BI'. L'esempio con cui si rende il carattere a larghezza fissa attraverso '@F' sembra contraddire questo, ma in realtà funziona perché si tratta di un comando riferito allo stile a cui poi si aggiunge un cambiamento di forma.

## Corpo

Il corpo del carattere può essere modificato all'interno del documento (con il comando '@Font' utilizzato in modo differente da quanto visto finora), oppure può essere dichiarato nel preambolo che descrive gli aspetti generali dello stile prescelto. In ogni caso si rap-

presenta attraverso un numero seguito dall'unità di misura. Di solito si fa riferimento a punti tipografici, 'p', dove per esempio '12p' rappresenta 12 punti tipografici (un punto = 1/72 di pollice).

Se il contesto lo consente, si possono indicare degli incrementi o delle riduzioni del valore precedente, dove per esempio '+2p' rappresenta l'incremento di due punti rispetto al carattere precedente e '-1p' rappresenta la riduzione di un punto. Nello stesso modo si possono indicare dei valori relativi, dove per esempio '1.5f' rappresenta una dimensione pari al 150 % del corpo utilizzato precedentemente.

Nel preambolo del documento si utilizza il comando '@InitialFont' che si trova in quasi tutti gli stili:

```
@InitialFont { stile forma corpo }
```

Nell'esempio introduttivo è stato utilizzato un carattere Times tondo chiaro da 24 punti:

```
@SysInclude { doc }
@Document
  @InitialFont { Times Base 24p }
  ..
  //
  @Text @Begin
```

Quando si vuole modificare il corpo del carattere all'interno del documento, si usa il comando '@Font', con una delle due forme seguenti:

```
corpo @Font { testo }
```

```
{ stile forma corpo } @Font { testo }
```

L'esempio seguente mostra in che modo agire per ridurre leggermente (di due punti) il corpo di una parola:

```
L'operatore -2p @Font { AND } restituisce il valore booleano...
```

La stessa cosa avrebbe potuto essere ottenuta delimitando l'indicazione del corpo attraverso le parentesi graffe.

```
L'operatore { -2p } @Font { AND } restituisce il valore booleano...
```

L'esempio seguente mostra invece come è possibile modificare lo stile, la forma e il corpo simultaneamente.

```
La parola chiave { Courier Bold -1p } @Font { AND } serve a...
```

In precedenza è stato mostrato l'uso del comando '@F' per ottenere un carattere dattilografico; per la precisione si ottiene un carattere Courier chiaro con una dimensione pari a un punto in meno rispetto al corpo circostante. In pratica, '@F' è equivalente al comando:

```
{ Courier Base -1p } @Font { testo }
```

## Display

Lout mette una cura particolare nella definizione di varie forme per fare risaltare un blocco di testo (*display*). Viene descritto brevemente l'elenco di quelle più comuni.

- Testo staccato.

```
@Display { testo }
```

```
@LeftDisplay { testo }
```

```
@LD { testo }
```

Il testo fornito come argomento del comando viene staccato in modo evidente dal paragrafo precedente e da quello successivo. La prima riga inizia senza rientri.

- Testo staccato e rientrato a sinistra.

```
@IndentedDisplay { testo }
```

```
@ID { testo }
```

Il testo fornito come argomento del comando viene staccato in modo evidente dal paragrafo precedente e da quello successivo, inoltre viene aumentato il margine sinistro.

- Testo staccato e rientrato a sinistra e a destra.

```
@QuotedDisplay { testo }
```

```
@QD { testo }
```

Il testo fornito come argomento del comando viene staccato in modo evidente dal paragrafo precedente e da quello successivo, inoltre viene aumentato il margine sinistro e anche quello destro.

## Caratteristiche interne dei paragrafi

In precedenza, in occasione della descrizione della struttura di un documento Lout, è stato descritto l'uso dei comandi di separazione dei paragrafi ('@LP', '@PP' e altri). Questi non servono a definire le caratteristiche interne ai paragrafi, che invece possono essere specificate attraverso alcuni comandi da collocare nel preambolo, oppure attraverso '@Break'.

Il comando '@Break' può essere utilizzato per intervenire in un paragrafo il cui contenuto gli viene fornito come argomento, ma a sua volta non può apparire da solo. In pratica, negli esempi mostrati, '@Break' viene posto come argomento di un ambiente *display* di qualche tipo.

### Interruzione e allineamento

Generalmente la suddivisione dei paragrafi in righe avviene in modo automatico, senza rispettare l'andamento del file sorgente. È possibile impedire la separazione in una certa posizione utilizzando il simbolo '~' in qualità di spazio orizzontale non interrompibile.

```
Il comando tar-cf-prova.tgz~/opt genera il file...
```

L'esempio mostra il modo in cui si può evitare che la descrizione di un comando del sistema operativo venga spezzato in corrispondenza degli spazi tra un argomento e l'altro.

Sia nel caso in cui la separazione in righe dei paragrafi venga ridefinita da Lout, sia quando si vogliono mantenere le interruzioni usate nel sorgente, si pone il problema di allineare il testo: a sinistra, in centro, a destra o simultaneamente a sinistra e a destra. Tutte queste cose si indicano attraverso una parola chiave che viene riconosciuta sia nel comando '@Break', sia nel comando '@InitialBreak' (il secondo si utilizza come opzione nel preambolo del documento). Tra queste si distinguono due gruppi importanti: quelle che terminano per 'ragged', che si riferiscono a righe ricomposte da Lout, e quelle che terminano per 'lines', che si riferiscono a righe interrotte esattamente come nel sorgente.

- 'ragged' -- allineamento normale a sinistra;
- 'cragged' -- allineamento centrato;
- 'rragged' -- allineamento a destra;
- 'adjust' -- allineamento simultaneo a sinistra e a destra;
- 'outdent' -- allineamento simultaneo a sinistra e a destra con la prima riga sporgente dal lato sinistro;
- 'lines' -- allineamento normale a sinistra rispettando le interruzioni di riga;
- 'clines' -- allineamento centrato rispettando le interruzioni di riga;

- `'rlines'` -- allineamento a destra rispettando le interruzioni di riga.

Distanza tra le righe

« La distanza tra le righe misura lo spazio che c'è tra la base di una riga e la base della successiva. Generalmente viene definito attraverso un valore relativo alla dimensione del carattere (al corpo), ma può essere indicato anche in modo assoluto. Per qualche motivo, il valore in questione deve essere terminato con il carattere `'x'`. Per esempio: `'1.20fx'` rappresenta una distanza di 1,2 volte il corpo del carattere (il 120 %); `'1.5vx'` rappresenta 1,5 volte la distanza preesistente, ma una notazione del genere può applicarsi solo quando esiste qualcosa di precedente a cui fare riferimento; `'14px'` rappresenta una distanza di 14 punti; `'1cx'` rappresenta una distanza di 1 cm.

Separazione in sillabe

« La separazione in sillabe è un procedimento che dipende dal linguaggio, cosa che normalmente si seleziona nel preambolo del documento attraverso il comando `'@InitialLanguage'`. L'attivazione o meno della sillabazione dipende dal comando `'@Break'` o da `'@InitialBreak'` nel preambolo, attraverso una parola chiave: `'hyphen'` per attivarla e `'nohyphen'` per disattivarla.

Quando la sillabazione è attivata, si può utilizzare il simbolo `'&-'` per indicare a Lout la posizione di una possibile separazione delle parole. Per esempio, `'hard&-ware'` fa sì che se necessario la parola possa essere separata esattamente alla metà.

Raccogliere tutto assieme

« Generalmente conviene regolare le caratteristiche dei paragrafi già nel preambolo, attraverso il comando `'@InitialBreak'`:

```
@InitialBreak { allineamento distanza hyphen|nohyphen }
```

L'esempio seguente ripropone quanto già visto in precedenza riguardo alla definizione di un documento in forma di libro. Si può osservare la scelta di indicare la distanza tra le righe come un valore relativo riferito al corpo del carattere utilizzato.

```
@SysInclude { book }
@Book
...
@InitialBreak { adjust 1.2fx hyphen }
...
//
```

All'interno del documento si può utilizzare il comando `'@Break'` nelle situazioni in cui ciò è possibile, per esempio in un ambiente che crea un blocco messo in risalto.

```
{ allineamento distanza hyphen|nohyphen } @Break { testo }
```

```
allineamento @Break { testo }
```

```
distanza @Break { testo }
```

```
hyphen|nohyphen @Break { testo }
```

L'esempio seguente mostra il caso di un paragrafo messo in risalto, nel quale viene ridotta la distanza tra le righe e si annulla la separazione in sillabe:

```
@QuotedDisplay { 0.8vx nohyphen } @Break {
Questa è un'informazione così importante
che facciamo in modo di rendervi difficile
la lettura }
```

Testo letterale

« Il testo letterale può essere indicato utilizzando l'ambiente del comando `'@Verbatim'`. Possono essere usate due modalità equivalenti, che però hanno risvolti diversi nel contenuto che può avere l'ambiente in questione.

```
@Verbatim { testo_letterale }
```

```
@Verbatim @Begin
testo_letterale
...
@End @Verbatim
```

Il primo dei due modi è adatto per le inclusioni brevi di testo, dove non si pongono problemi nell'uso delle parentesi graffe (queste possono essere contenute nel testo letterale, ma devono essere bilanciate correttamente); il secondo è l'alternativa per i blocchi di testo più lunghi e per quelle situazioni in cui le parentesi graffe possono creare dei problemi.

Se si utilizza il secondo modo di inclusione di testo letterale, il testo in questione non può contenere la parola `'@End'`.

Spesso, il testo incluso in modo letterale viene reso con un carattere dattilografico, come nell'esempio seguente o in quello successivo.

```
{ Courier Base } @Font @Verbatim @Begin
$ ls -l <invio>
...
@End @Verbatim
```

```
@F @Verbatim @Begin
$ ls -l <invio>
...
@End @Verbatim
```

Elenchi

« Gli elenchi di Lout sono molto sofisticati, permettendo di gestire sia degli elenchi semplici, sia gli elenchi descrittivi, sia una lunga serie di elenchi puntati o numerati in vario modo. Tutti gli elenchi di Lout hanno in comune il simbolo che serve a concludere l'ambiente dell'elenco: `'@EndList'`.

Gli elenchi semplici sono solo un modo per staccare il testo evidenziandone così gli elementi. Questo si ottiene con l'ambiente introdotto dal simbolo `'@List'`:

```
@List
@ListItem elemento
@ListItem elemento
...
@EndList
```

Il risultato che si ottiene è costituito da una serie di paragrafi, uno per ogni punto, rientrati a sinistra e staccati verticalmente più dei paragrafi normali. L'esempio seguente dovrebbe rendere meglio l'idea.

```
@List
@ListItem { Tizio Tizi }
@ListItem {
Caio Cai, nato a Sferopoli il giorno -- e trasferitosi
in altra città in seguito a.. }
@ListItem { @B Sempronio Semproni }
@EndList
```

Gli elenchi puntati si ottengono con gli ambienti introdotti da uno tra i simboli `'@BulletList'`, `'@StarList'` e `'@DashList'`. Si tratta rispettivamente di elenchi le cui voci sono precedute da un punto, un asterisco o un trattino.

```
@BulletList | @StarList | @DashList
@ListItem elemento
@ListItem elemento
...
@EndList
```

Il risultato che si ottiene è lo stesso dell'elenco semplice, con l'aggiunta del puntino (o dell'asterisco o del trattino) nella parte sinistra all'inizio delle voci. L'esempio seguente è una variante di quello già presentato per l'elenco semplice, dove l'inizio delle voci è asteriscato.

```
@StarList
@ListItem { Tizio Tizi }
@ListItem {
  Caio Cai, nato a Sferopoli il giorno ... e trasferitosi
  in altra città in seguito a- }
@ListItem { @B Sempronio Semproni }
@EndList
```

Gli elenchi numerati si ottengono con gli ambienti introdotti da uno tra i simboli '@NumberedList', '@RomanList', '@UCRomanList', '@AlphaList', '@UCAAlphaList', e dalla serie parallela '@ParenNumberedList', '@ParenRomanList', '@ParenUCRomanList', '@ParenAlphaList', '@ParenUCAAlphaList'.

I due raggruppamenti di simboli Lout si riferiscono a numerazioni normali o numerazioni tra parentesi ('Paren'); i simboli il cui nome contiene la parola 'Roman' rappresentano una numerazione romana; i simboli il cui nome contiene la parola 'Alpha' rappresentano una numerazione alfabetica; il prefisso 'UC' specifica che si tratta di una numerazione (romana o alfabetica) maiuscola.

```
@NumberedList | @RomanList | @UCRomanList | @AlphaList |
@UCAAlphaList
@ListItem elemento
@ListItem elemento
...
@EndList
```

```
@ParenNumberedList | @ParenRomanList | @ParenUCRomanList ↔
↔ | @ParenAlphaList | @ParenUCAAlphaList
@ListItem elemento
@ListItem elemento
...
@EndList
```

Per la realizzazione di elenchi composti, dove un punto si articola in sottopunti, basta inserire un elenco all'interno di una voce, per esempio nel modo seguente:

```
@NumberList
@ListItem { Tizio Tizi }
@ListItem { Caio Cai
  @BulletList
  @ListItem { nato a Sferopoli il... }
  @ListItem { residente a- }
  @EndList }
@ListItem { @B Sempronio Semproni }
@EndList
```

Gli elenchi descrittivi permettono di specificare ciò che si vuole usare per indicare ogni voce. In pratica si tratta di una stringa che rappresenta un'etichetta, ovvero una sorta di titolo della voce. Lout mette a disposizione diversi simboli in funzione della distanza che si intende lasciare tra l'inizio dell'etichetta e il blocco di testo a cui questa fa riferimento: '@TaggedList', '@WideTaggedList' e '@VeryWideTaggedList'.

```
@TaggedList | @WideTaggedList | @VeryWideTaggedList
@TagItem { etichetta } { elemento }
@TagItem { etichetta } { elemento }
...
@EndList
```

Quando per qualche motivo si ha a che fare con etichette troppo lunghe, o comunque può risultare inopportuno fare iniziare il blocco di testo sulla stessa riga dell'etichetta, al posto del simbolo '@TagItem' per introdurre le voci si può usare '@DropListItem'.

```
@TaggedList | @WideTaggedList | @VeryWideTaggedList
@DropListItem { etichetta } { elemento }
@DropListItem { etichetta } { elemento }
...
@EndList
```

## Note

Lout organizza in modo molto raffinato le note a piè pagina, le note finali e le note a margine. Qui vengono mostrate solo le caratteristiche essenziali.

Note a piè pagina e note finali

La distinzione tra note a piè pagina e note finali sta nel fatto che le prime appaiono nella stessa pagina in cui si trova il loro riferimento, o al massimo in quella successiva, mentre le seconde si collocano alla fine del documento (o alla fine del capitolo).

```
@FootNote [ @Location { ColFoot | PageFoot } ] { testo }
```

```
@EndNote { testo }
```

Come si può intuire, il comando '@FootNote' riguarda l'inserimento di una nota a piè pagina, mentre '@EndNote' di una nota alla fine del documento. In particolare, la nota a piè pagina può essere collocata alla fine della colonna, o alla fine della pagina; in questo ultimo caso può occupare tutte le colonne della pagina. Utilizzando l'opzione '@Location' con l'argomento 'ColFoot' si ottiene una nota che si espande orizzontalmente solo nello spazio della colonna, mentre con 'PageFoot', si vuole fare in modo che la nota si allarghi per tutto lo spazio orizzontale della pagina.

```
@FootNote
  @Location { PageFoot }
{ Questa è una nota a piè pagina che si espande
  orizzontalmente occupando tutta la larghezza della
  pagina, anche se questa è suddivisa in più di una
  colonna. }
```

```
@EndNote { Questa è una nota alla fine del
  documento. }
```

In generale, è sconsigliabile l'uso simultaneo di note a piè pagina e note a fine documento, in quanto non è possibile distinguere facilmente i riferimenti che vengono collocati nella composizione finale.

Note a margine

Le note a margine sono annotazioni molto brevi che si collocano sullo spazio del margine sinistro o del margine destro della pagina. Per ottenerle si utilizzano i comandi '@LeftNote' o '@RightNote' a seconda che si voglia la nota sul margine sinistro o sul margine destro. Se si inseriscono in un documento che distingue tra pagine destre e sinistre, si possono utilizzare i comandi '@OuterNote' e '@InnerNote' per indicare rispettivamente le note sul margine esterno o sul margine interno.

```
@LeftNote { nota_sul_margine_sinistro }
```

```
@RightNote { nota_sul_margine_destro }
```

```
@InnerNote { nota_sul_margine_interno }
```

```
@OuterNote { nota_sul_margine_esterno }
```

## Figure e tabelle fluttuanti

« Come per LaTeX, le figure e le tabelle possono essere parte del testo normale, oppure possono essere inserite in un involucro che le rende fluttuanti. L'involucro in questione è praticamente identico nei due casi, a parte il simbolo iniziale che serve a distinguere la numerazione delle figure da quella delle tabelle. Di conseguenza, l'oggetto che compone la figura o la tabella all'interno di questo involucro, può essere qualunque cosa, in base alle intenzioni dell'autore.

```
@Figure | @Table
  @Location { collocazione }
  @OnePage { Yes | No }
  @FullPage { Yes | No }
  @CaptionPos { Above | Below }
  @Caption { didascalia }
oggetto
```

Come accennato, il simbolo '@Figure' rappresenta un involucro fluttuante per una figura, mentre '@Table' è quello da usare per una tabella. Le opzioni che si vedono nello schema sintattico sono tutte facoltative:

- il simbolo '@Location' permette di definire la collocazione fluttuante della figura o della tabella, attraverso un argomento composto da una parola chiave:
  - 'PageTop' -- l'oggetto deve essere collocato all'inizio della pagina successiva,
  - 'PageFoot' -- l'oggetto deve essere collocato alla fine della pagina corrente,
  - 'ColTop' -- l'oggetto deve essere collocato all'inizio della colonna successiva,
  - 'ColFoot' -- l'oggetto deve essere collocato alla fine della colonna corrente,
  - 'ColEnd' -- l'oggetto deve essere collocato in una colonna alla fine del documento (o del capitolo),
  - 'AfterLine' -- l'oggetto deve essere collocato esattamente dove si trova (nella riga successiva in base al risultato della composizione),
  - 'TryAfterLine' -- l'oggetto deve essere collocato esattamente dove si trova, a meno che lo spazio sia insufficiente, perché in tal caso viene spostato all'inizio della colonna successiva,
  - 'Display' -- l'oggetto deve essere messo in risalto e collocato esattamente dove si trova,
  - 'Raw' -- l'oggetto non deve essere fluttuante e deve rimanere com'è (serve a inserire delle immagini all'interno delle celle di una tabella);
- il simbolo '@OnePage' permette di definire se si vuole che l'oggetto fluttuante debba rimanere intero o se possa essere diviso tra una pagina e la successiva (o tra colonne), il valore predefinito varia in funzione del tipo di collocazione prescelto;
- il simbolo '@FullPage' permette di definire se si vuole che l'oggetto fluttuante debba occupare da solo lo spazio di una pagina, oppure se questo possa essere condiviso con il testo;
- il simbolo '@Caption' permette di indicare la didascalia dell'oggetto;
- il simbolo '@CaptionPos' permette di stabilire la posizione in cui deve apparire la didascalia (in alto, 'Above', o in basso, 'Below').

## Figure

« Come già spiegato, qualunque oggetto può essere una figura. Di solito questo oggetto è ottenuto con il comando '@Fig', che qui non viene descritto. In alternativa si mostra in che modo inserire del testo letterale, che alle volte può servire per lo scopo. Si osservi l'esempio seguente:

```
@Figure
  @Location { TryAfterLine }
  @OnePage { Yes }
  @FullPage { No }
  @CaptionPos { Below }
  @Caption { @I { standard input } e @O { standard output } }
  @F @Verbatim @Begin
STDIN ----->| Programma |-----> STDOUT
                |          |-----> STDERR
                |          |
  @End @Verbatim
```

La figura dell'esempio rappresenta uno schema costruito attraverso del testo letterale utilizzando un carattere dattilografico. Questa dovrebbe essere collocata immediatamente sotto il punto in cui appare nel sorgente; se non dovesse esserci spazio sufficiente fino alla fine della pagina, verrebbe spostata all'inizio di quella successiva.

Nel caso si realizzino figure nel modo proposto dall'esempio, occorre fare attenzione a non inserire delle tabulazioni nel sorgente, perché verrebbero interpretate in un modo diverso da quello che può fare il programma che si utilizza per la sua scrittura. Eventualmente si può filtrare il sorgente con il comando '**expand -8**', in modo da ottenere la trasformazione delle tabulazioni in spazi normali.

## Tabelle

« Le tabelle di Lout possono essere molto sofisticate. Anche in questo caso vale lo stesso discorso fatto per le figure, dove l'oggetto che si intende fornire per la descrizione della tabella può essere qualsiasi cosa, anche se in pratica si tratta quasi sempre di un comando '@Tab'. Qui si intende mostrare solo un uso elementare; dettagli maggiori possono essere trovati nella documentazione originale.

Per poter utilizzare le tabelle di Lout, cioè quelle che si ottengono con il comando '@Tab', occorre includere uno stile aggiuntivo prima della dichiarazione del tipo di documento fondamentale:

```
@SysInclude { tab }
```

Per esempio, nel caso del documento ordinario si dovrebbe iniziare nel modo seguente:

```
@SysInclude { tab }
@SysInclude { doc }
@Document
...
//
```

Quello che si vede sotto è lo schema sintattico di una tabella estremamente semplificata. Si noti in particolare il fatto che le colonne sono distinte da una lettera alfabetica maiuscola.

```
@Tab
  @Fmta { @Col A ! @Col B ! ... }
{
  @Rowa A { cella_1.1 } B { cella_1.2 } ...
  @Rowa A { cella_2.1 } B { cella_2.2 } ...
  ...
}
```

Utilizzando questo schema semplificato, si ottengono delle tabelle senza linee (né verticali, né orizzontali) per evidenziare o abbellire le sue parti. Di solito, per quanto semplice sia la tabella, si ha almeno l'esigenza di utilizzare delle linee orizzontali per evidenziare le righe che compongono l'intestazione delle colonne e per segnalare la fine della tabella. Per questo si devono usare delle parole chiave aggiuntive che si collocano tra gli argomenti di '@Rowa' cioè dei comandi che descrivono le righe.

```
@Rowa
above { single }
A { cella } B { cella } ...
below { single }
```

L'opzione 'above { single }' inserisce una linea orizzontale sopra la riga a cui si riferisce, mentre 'below { single }' la inserisce sotto. Supponendo di voler ottenere una tabella come quella schematizzata qui sotto,

Parametro LOC	Posizione corrispondente
h	posizione attuale
t	superiore
b	inferiore
p	pagina

Esempio di tabella.

il codice necessario per Lout potrebbe essere quello seguente:

```
@Table
@Location { TryAfterLine }
@OnePage { Yes }
@FullPage { No }
@CaptionPos { Below }
@Caption { Esempio di tabella }
@Tab
@Fmta { @Col A ! @Col B }
{
@Rowa
above { single }
A { Parametro LOC }
B { Posizione corrispondente }
below { single }
@Rowa
A { h }
B { posizione attuale }
@Rowa
A { t }
B { superiore }
@Rowa
A { b }
B { inferiore }
@Rowa
A { p }
B { pagina }
below { single }
}
```

## Indici e riferimenti incrociati

Lout crea automaticamente una serie di riferimenti incrociati. I più comuni sono quelli dei piè pagina e quelli degli indici. Lout richiede l'elaborazione ripetuta di un sorgente per sistemare proprio questi indicatori; in particolare, a differenza di LaTeX (che in generale richiede tre passaggi, o quattro se si inserisce BibTeX), non si può prevedere quante volte debba essere rifatta la composizione.

### Riferimenti nel testo

Come accennato, le note a piè pagina e quelle alla fine del documento sono un esempio di inserzione nel testo di riferimenti a qualcosa che appare altrove. Quando si vuole indicare un riferimento a qualcosa di diverso, si usano i comandi '@PageOf' e '@NumberOf'.

```
@PageOf { nome_del_riferimento }
```

```
@NumberOf { nome_del_riferimento }
```

Il primo dei due viene rimpiazzato da Lout con il numero della pagina in cui si trova il riferimento indicato, mentre il secondo mostra il numero del capitolo o della sezione relativa. Per esempio, se da qualche parte è stato dichiarato il riferimento denominato 'presentazione', vi si può fare riferimento come in questo estratto:

```
Come accennato in precedenza (a pagina @PageOf { presentazione } ),
la matematica non è un'opinione.
```

Successivamente, il testo che si vede sopra si trasforma nella composizione in qualcosa di simile a quello che segue:

```
Come accennato in precedenza (a pagina 11),
la matematica non è un'opinione.
```

La dichiarazione di un riferimento (in altri termini di un'etichetta) può essere fatta con il comando '@PageMark', oppure con l'opzione '@Tag' che si può inserire all'inizio dei capitoli, delle sezioni, delle appendici e delle loro strutture inferiori.

```
@PageMark { nome_del_riferimento }
```

```
@Chapter | @Section | @SubSection | @SubSubSection
@Title { titolo }
@Tag { nome_del_riferimento }
...
@Begin
...
```

Riprendendo l'esempio precedente, si vede come potrebbe essere dichiarato un riferimento raggiungibile attraverso il comando '@PageOf':

```
Attenzione: @MarkOf { presentazione } la matematica
non è un'opinione perché...
```

Naturalmente, nel testo risultante dalla composizione non si vede la dichiarazione.

Lout è un po' rigido nell'uso di questi riferimenti: attraverso '@PageOf' si può fare riferimento alla pagina che contiene sia un riferimento dichiarato con '@PageMark', sia un riferimento dichiarato all'inizio della struttura per mezzo dell'opzione '@Tag', ma con '@NumberOf' si può solo fare riferimento solo a quanto dichiarato con l'opzione '@Tag'.

I nomi utilizzati per indicare i riferimenti devono essere univoci. Generalmente si utilizzano nomi composti solo da lettere alfabetiche ed eventualmente dal punto (come suggerisce l'autore di Lout). Se ce ne fosse la necessità, si può sempre delimitare questi nomi attraverso l'uso delle virgolette.

### Indice generale e indice analitico

A seconda dello stile del documento prescelto, l'indice generale viene incluso o meno, in modo automatico. Questo comportamento può essere modificato ritoccando il file di stile, oppure, creando uno stile personalizzato. Gli stili standard prevedono al massimo la stampa dell'indice generale; se si desidera ottenere un indice delle figure o delle tabelle occorre intervenire nello stile in ogni caso.

Anche l'indice analitico viene aggiunto automaticamente se il tipo di documento è adatto per questo, però in tal caso dipende dall'autore l'inserimento dei riferimenti che lo generano. Lout consente l'uso di una grande varietà di tecniche per ottenere un indice analitico veramente buono. Qui viene mostrato l'essenziale.

```
chiave_per_ordinamento @Index { voce }
```

Quello che si vede è la sintassi minima per inserire un riferimento nel testo che poi si traduce in una voce nell'indice analitico. La voce in questione viene mostrata utilizzando quanto indicato alla destra del simbolo '@Index', ordinata in base alla chiave indicata alla sua sinistra.

Il principio è che l'ordine (alfabetico) con cui devono essere ordinate le voci potrebbe essere diverso da quello che si viene a generare utilizzando direttamente il contenuto delle voci. Per fare un esempio tra le tante situazioni che si possono creare, la voce 'Decimo' potrebbe dover apparire prima di 'De Tizi', ma utilizzandole così come sono, lo spazio tra 'De' e 'Tizi' farebbe sì che questa ultima voce appaia per prima. Per questo è necessario specificare una voce alternativa da utilizzare per l'ordinamento. Per convenzione, oltre che per evitare imprevisti, è bene limitarsi all'uso delle sole lettere alfabetiche minuscole, non accentate, ed è per questo che nella sintassi non sono state usate le parentesi graffe per racchiudere l'argomento a sinistra del simbolo '@Index'.

Volendo realizzare un indice analitico strutturato in voci e sotto-voci, si può utilizzare il comando '@SubIndex', con l'aggiunta del comando 'SubSubIndex' per una suddivisione ulteriore.

```
chiave_per_ordinamento @SubIndex { voce }
```

```
chiave_per_ordinamento @SubSubIndex { voce }
```

Le sotto-voci sono interessanti in quanto riferite a una voce di livello precedente. La documentazione di Lout suggerisce di utilizzare delle chiavi strutturate, ottenute a partire dalla chiave della voce principale unendo un punto e aggiungendo un'estensione opportuna.

```
Tizio Tizi tiziotizi @Index { Tizio Tizi } è stato lo
scopritore di...
...
Tizio Tizi tiziotizi.origini @SubIndex { origini } era figlio di
Pinco Pallino e di...
```

L'esempio dovrebbe mostrare in maniera sufficientemente chiara il concetto: da qualche parte del testo si parla di 'Tizio Tizi' e lì viene inserito un riferimento; da un'altra parte si parla sempre di lui, ma in particolare si descrivono le sue origini. In pratica, la voce 'origini' dipende da 'Tizio Tizi' e opportunamente la chiave di ordinamento fa in modo che questa risulti successiva.

Seguendo la logica dell'esempio mostrato, se si scrive un capitolo su 'Tizio Tizi', potrebbe non avere significato un riferimento a una pagina in cui si parla di questa persona, mentre ci si troverebbe ad avere solo delle sottoclassificazioni (origini, vita, morte, ecc.). Volendo indicare una voce senza che con questa si ottenga il numero della pagina corrispondente, si può utilizzare il comando '@RawIndex'.

```
chiave_per_ordinamento @RawIndex { voce }
```

L'esempio già mostrato potrebbe essere modificato convenientemente nel modo seguente:

```
Tizio Tizi tiziotizi @RawIndex { Tizio Tizi } è stato lo
scopritore di...
...
Tizio Tizi tiziotizi.origini @SubIndex { origini } era figlio di
Pinco Pallino e di...
...
Tizio Tizi è nato tiziotizi.nascita @SubIndex { nascita }
in un paesino sperduto...
...ed è morto tiziotizi.morte @SubIndex { morte } il giorno
...
```

Problemi connessi alla generazione dei riferimenti incrociati

Quando si modifica un documento che fa uso di riferimenti incrociati di qualunque tipo (praticamente sempre), prima di riavviare l'eseguibile 'lout' per ottenerne la composizione sarebbe opportuno eliminare i file transitori che vengono creati da questo. Supponendo di lavorare con il file 'pippo', occorrerebbe eliminare il file 'pippo.ld' e 'lout.li'.

Diversamente, è probabile che una sola passata basti a ottenere il formato finale senza ottenere segnalazioni di errore, ma i riferimenti aggiunti nel documento potrebbero essere errati o mancare del tutto.

## Localizzazione

I problemi di localizzazione di un documento riguardano generalmente le definizioni standard di alcune componenti tipiche (capitolo, appendice, indice, ecc.) e la sillabazione. Per attuare questo con Lout si utilizza l'opzione '@InitialLanguage' nel preambolo del documento, mentre a livelli inferiori si possono circoscrivere delle eccezioni.

```
@SysInclude { book }
@Book
...
@InitialLanguage { Italian }
...
//
```

L'esempio mostra in che modo potrebbe essere definito il linguaggio «italiano» per tutto un documento (in questo caso un libro).

All'interno del testo è possibile alterare il linguaggio generale attraverso il comando '@Language':

```
linguaggio @Language { testo }
```

Per esempio, per indicare che una frase è scritta in tedesco si potrebbe fare come nell'esempio seguente:

```
La nonna disse: German @Language { Wer bekommt die Torte? }
```

Configurazione di una localizzazione

Nel momento in cui viene scritto questo capitolo, le versioni di Lout che si trovano comunemente in circolazione non dispongono del linguaggio italiano. Per prepararselo occorre intervenire su alcuni file: '/usr/lib/lout/include/langdefs', '/usr/lib/lout/data/standard.ld' e '/usr/lib/lout/hyph/italian.lh'. L'ultimo di questi serve per definire le regole della sillabazione e di solito viene creato a partire da quello di un altro paese. Questo file è diviso in due parti, dove la seconda, cioè quella che indica precisamente le regole della separazione in sillabe, può essere ricopiata dal file corrispondente utilizzato per LaTeX.

Personalizzazione dello stile

Invece di utilizzare uno degli stili standard di Lout, si può creare il proprio, di solito modificandone uno preesistente. Quando si crea uno stile riferito a un documento particolare, può darsi che il file relativo venga tenuto assieme a quello del documento stesso; in tal caso può convenire di utilizzare un comando di inclusione diverso dal solito. Supponendo di voler creare una variante dello stile 'book', si potrebbe copiare il file corrispondente, '/usr/lib/lout/include/book', nella directory di lavoro del documento e chiamarlo 'libro'. In questo modo, l'inizio del documento potrebbe essere organizzato nel modo seguente:

```
@Include { libro }
@Book
//
@Chapter @Begin
...
```

Si osservi l'uso del comando '@Include' che si riferisce alla directory corrente o a un percorso assoluto (se indicato).

Nelle sezioni seguenti si accenna all'organizzazione di questo file di stile. Per modificarlo basta intervenire negli argomenti delle opzioni indicate; anche senza conoscere precisamente i dettagli, si dovrebbe riuscire nell'intento utilizzando semplicemente l'intuito.

Inclusione di altri stili

Nella prima parte del file di stile si incontra una serie di inclusioni possibili per l'aggiunta di altri stili.

```
#####
#
# @SysInclude commands for standard packages.
#
#####
@SysInclude { fontdefs } # font definitions
@SysInclude { langdefs } # language definitions
@SysInclude { dl } # DocumentLayout package
@SysInclude { bookf } # BookLayout extension
# @SysInclude { tab } # @Tab table formatter
# @SysInclude { eq } # @Eq equation formatter
# @SysInclude { fig } # @Fig advanced graphics
# @SysInclude { graph } # @Graph graph drawing
# @SysInclude { cprint } # @CPrint C and C++ programs
# @SysInclude { pas } # @Pas Pascal programs
```

Come si vede, le inclusioni che non sono necessarie appaiono commentate. Potrebbe essere conveniente togliere il commento da qualcosa, per esempio l'inclusione dello stile 'tab' in modo da consentire la realizzazione di tabelle attraverso il comando '@Tab'.

Dopo le inclusioni standard appare l'inserimento predefinito dello stile 'mydefs', nel caso fosse presente nella directory di lavoro.

ro nel momento della composizione. In pratica, questo è il nome convenzionale di un file da usare per la personalizzazione aggiuntiva.

```
#####  
#  
# @Include command for reading personal definitions from current directory. #  
#  
#####  
  
@Include { mydefs }
```

### Veste grafica del documento

Nell'ultima parte del file di stile si definisce una serie di cose che riguardano la veste grafica del documento. Nei file di configurazione standard sono riportate tutte le opzioni disponibili con gli argomenti predefiniti, commentate attraverso il carattere '#' e descritte.

```
@Use { @DocumentLayout  
# @InitialFont { Times Base 12p } # initial font  
# @InitialBreak { adjust 1.20fx hyphen } # initial break  
# @InitialSpace { lout } # initial space style  
# @InitialLanguage { English } # initial language  
...  
}
```

### Particolarità del tipo di documento

A seconda dello stile originale da cui si è partiti per realizzare il proprio, l'ultima parte potrebbe essere diversa. Per esempio, nel caso del libro, questa comincia così:

```
@Use { @BookLayout  
# @TitlePageFont { Helvetica Base } # title page font (not size)  
# @SeparateIntroNumbering { Yes } # separate intro page numbers  
# @ChapterStartPages { Any } # Any, Odd, or Even  
# @ReferencesBeforeAppendices { No } # pos of ref list  
...  
}
```

### Parte conclusiva

La parte finale del file della configurazione dello stile viene lasciato normalmente così come si trova.

```
#####  
#  
# @Database (and @SysDatabase) clauses go here. #  
#  
#####  
  
@SysDatabase @RefStyle { refstyle } # reference printing styles
```

### Riferimenti

- Jeffrey H. Kingstom, *A User's Guide to the Lout Document Formatting System*
- Jeffrey H. Kingstom, *A Practical Introduction to the Lout Document Formatting System*

<sup>1</sup> Lout GNU GPL

## Introduzione a LyX

Creazione di un documento .....	705
Struttura e stile .....	706
Modelli di documento .....	707
Automatismi .....	707
Riferimenti .....	707

LyX <sup>1</sup> è un sistema di composizione tipografica visuale, che si avvale principalmente di LaTeX per generare il risultato finale. Sono disponibili almeno due versioni di LyX differenti, in base al tipo di librerie grafiche utilizzate: Qt oppure XForms (nel secondo caso si tratta di software proprietario).

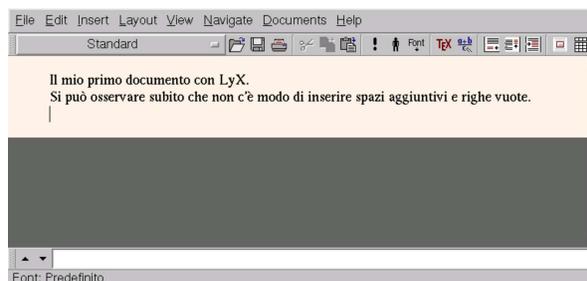
LyX viene definito dai suoi sviluppatori come un sistema di scrittura WYSIWYM, ovvero, *What you see is what you mean*, a differenza dei sistemi visuali comuni (definiti WYSIWYG, ovvero *What you see is what you get*), perché ciò che si vede sullo schermo dà solo l'idea del risultato finale.

In generale, si può considerare LyX come il tramite per coloro che sono spaventati dallo scrivere un documento senza l'aiuto di uno strumento visuale, anche se da un punto di vista operativo, alla fine, la scrittura diretta di un sorgente (LaTeX, Lout, SGML, XML o altro), è sempre la scelta migliore. In questo modo ci si può abituare all'idea e poi il passaggio è meno traumatico. Questa precisazione è bene farla, perché non ci si può aspettare da LyX la stabilità di funzionamento che si può avere scrivendo direttamente un sorgente per la composizione differita; pertanto diventa difficile trovare delle motivazioni migliori a quella espressa per usare LyX piuttosto di un altro sistema di scrittura visuale.

### Creazione di un documento

Si avvia LyX con l'eseguibile 'lyx', che può essere avviato senza argomenti. Per creare un documento nuovo, basta aprire il menù *File* e selezionare la voce *New*. La figura u94.1 mostra in che modo si può presentare LyX mentre si scrive un file per la prima volta. Mentre si scrive, si può osservare subito che non si possono inserire spazi aggiuntivi e nemmeno righe vuote.

Figura u94.1. Scrittura di un documento nuovo con LyX.



Si può salvare il documento selezionando la voce *Save as* dal menù *File*, specificando poi il nome del file, che ha preferibilmente l'estensione '.lyx'.

Il file che si ottiene è un file di testo che ha una vaga somiglianza con TeX. Quanto si vede nella figura u94.1 si traduce in pratica nel testo seguente:

```
#LyX 1.3 created this file. For more info see http://www.lyx.org/  
\lyxformat 221  
\textclass article  
\language english  
\inputencoding auto  
\fontscheme default  
\graphics default  
\paperfontsize default  
\papersize Default  
\paperpackage a4  
\use_geometry 0  
\use_amsmath 0  
\use_natbib 0
```

```

\use_numerical_citations 0
\paperorientation portrait
\secnumdepth 3
\tocdepth 3
\paragraph_separation indent
\defekip medskip
\quotes_language english
\quotes_times 2
\papercolumns 1
\papersides 1
\paperpagestyle default

\layout Standard

Il mio primo documento con LyX.
\layout Standard

Si può osservare subito che non c'è modo di inserire spazi aggiuntivi e
righe vuote.
\layout Standard

\the_end

```

Il documento può essere stampato selezionando la voce *Print* dal menù *File*. Per arrivare alla stampa, LyX usa LaTeX in modo trasparente, con un file che potrebbe essere simile a quello seguente:

```

%% LyX 1.3 created this file. For more info, see http://www.lyx.org/.
%% Do not edit unless you really know what you are doing.
\documentclass[english]{article}
\usepackage[T1]{fontenc}
\usepackage[latin1]{inputenc}

\makeatletter

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% LyX specific LaTeX commands.
\providecommand{\LyX}{L\kern-.1667em\lower.25em\hbox{Y}\kern-.125em\@}

\usepackage{babel}
\makeatother
\begin{document}
Il mio primo documento con \LyX{}.

Si può osservare subito che non c'è modo di inserire spazi aggiuntivi
e righe vuote.
\end{document}

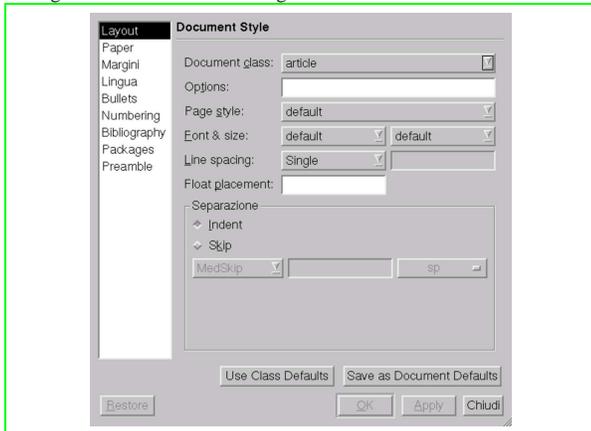
```

Di conseguenza, ciò che si ottiene con la stampa è esattamente il risultato della composizione di questo file con LaTeX.

### Struttura e stile

Dal menù *Layout* è possibile accedere a funzionalità che cambiano l'aspetto del documento. La cosa più importante che si deve stabilire del documento che si va a scrivere è la sua struttura, secondo dei modelli prestabiliti, attraverso la voce *Document*. La figura u94.4 mostra la finestra di inserimento che si ottiene, dove si può osservare in alto la presenza di un menù a scomparsa con un elenco di stili generali.

Figura u94.4. Caratteristiche generali del documento.

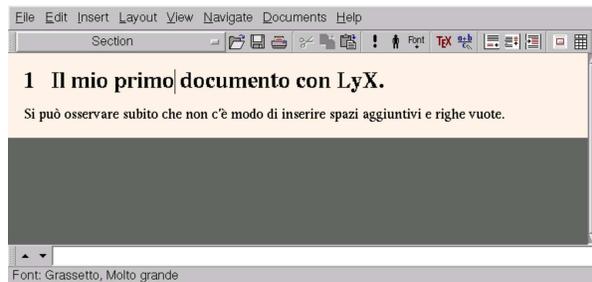


Selezionando uno stile complessivo differente, la struttura del documento cambia. Per esempio, il tipo di documento può ammettere una suddivisione in capitoli, oppure solo in sezioni di minore importanza, oppure può essere sprovvisto di suddivisioni. Ancora più importante di questo, la scelta dello stile complessivo dichiara anche il modo in cui si vuole arrivare alla composizione finale. Nell'e-

sempio iniziale della figura u94.1 si fa riferimento allo stile generale predefinito, corrispondente allo stile *'article'* di LaTeX; in questo modo la composizione passa per la trasformazione in LaTeX. Se si osservano le voci che appaiono nel menù a scomparsa della voce *Layout* già descritta (figura u94.4), si possono notare alcuni stili riferiti espressamente a DocBook: il loro utilizzo implica una composizione che utilizza strumenti relativi a DocBook, pertanto viene generato un sorgente SGML e non più LaTeX.

Durante la scrittura del documento, il testo che viene inserito viene associato a uno stile predefinito, in relazione al tipo di documento scelto. Se si guarda nuovamente la figura u94.1, si può notare che sotto alle voci del menù normale appare un menù a scomparsa, sul quale si legge il nome *'standard'*. Il significato è semplice: il testo sul quale si trova il cursore è associato allo stile standard. Basta mettere il cursore su un blocco di testo, per esempio la prima riga, scegliere una voce differente da questo menù a scomparsa per ottenerne l'adattamento al suo stile. Nella figura u94.5 è stato associato il testo della prima riga allo stile *'section'*, trasformandolo così nel titolo di una sezione.

Figura u94.5. Modifica dello stile associato a un blocco di testo.



Naturalmente, nell'ambito delle possibilità dello stile a cui è associato il blocco di testo, è possibile intervenire per modificare localmente una porzione di questo o anche il paragrafo nel suo complesso, ma per questo si deve agire sempre nel menù *Layout*, selezionando altre voci.

### Modelli di documento

LyX consente di creare un documento a partire da un modello, utilizzando la voce *New from template* del menù *File*. In pratica, si tratta soltanto di file LyX di esempio che si trovano in una posizione conveniente del file system, che l'utente comune non può modificare.

### Automatismi

Quando il testo prevede una numerazione, come avviene con i titoli delle sezioni, i riferimenti in nota, i riferimenti alle pagine di un indice, tutto avviene in modo automatico. Tuttavia, si tratta di un compito demandato al sistema di composizione (LaTeX, DocBook o altro). L'inserimento di oggetti di questo genere nel testo, si ottiene con le voci del menù *Insert*. Per esempio, per inserire un indice generale, si deve selezionare la voce *Lists & TOC*, quindi *Table of contents*.

### Riferimenti

- *The document processor*  
<http://www.lyx.org>

<sup>1</sup> LyX GNU GPL

Installazione .....	709
Sesh .....	710
Utilizzare HieroTeX .....	710
Due modi di usare HieroTeX .....	711
Scrittura normale .....	711
Codifica di HieroTeX .....	714
Riferimenti .....	720

HieroTeX è un sistema per la composizione con caratteri geroglifici attraverso LaTeX. Si compone di una serie di file di stile e una serie di file di caratteri tipografici; inoltre fornisce alcuni programmi di servizio, in particolare Sesh, il cui scopo è quello di filtrare un file LaTeX per comporre le istruzioni corrette per la generazione di un testo in geroglifico.

Probabilmente non esiste alcun pacchetto già pronto per la propria distribuzione GNU/Linux e occorre fare da soli: sia l'installazione degli stili e dei caratteri, sia la compilazione di Sesh.

Lo scopo di questo capitolo è solo quello di mostrare come si usa HieroTeX. Chi scrive queste informazioni non ha alcuna preparazione su tale forma di scrittura: l'unica motivazione da cui è nato questo capitolo è la curiosità. È probabile che in queste pagine appaiono degli esempi senza senso nella lingua dell'antico Egitto, cosa di cui deve tenere conto il lettore.

HieroTeX può essere ottenuto dal sito gestito dal suo stesso autore, Serge Rosmorduc e precisamente dall'URI <http://www.iut.univ-paris8.fr/~rosmord/archives/>, prelevando i file corrispondenti ai modelli: 'egyptomf-\*.tar.gz', 'egyptopk-\*.tar.gz', e 'egyptouser-\*.tar.gz'.

## Installazione

Dopo aver prelevato i tre file indicati all'inizio, si estrae il loro contenuto, così si ottiene la directory 'HieroTeX/' a partire da quella corrente.

```
tar xzvf egypto--tar.gz
```

La prima cosa da fare è installare i caratteri tipografici e gli stili per TeX. Dal momento che ogni distribuzione GNU/Linux è organizzata a modo suo, per quanto riguarda TeX, bisogna fare una piccola ricerca per determinare dove sono stati collocati gli altri. Occorre cercare la posizione dei file '\*.mf', '\*.tfm' e '\*.sty'. A titolo di esempio, potrebbe trattarsi delle directory '/usr/share/texmf/fonts/source/pacchetto\_tex' per i file '\*.mf', della directory '/usr/share/texmf/fonts/tfm/pacchetto\_tex' per i file '\*.tfm' e della directory '/usr/share/texmf/tex/latex/pacchetto\_tex' per gli stili. In tal caso, si potrebbe procedere come viene mostrato di seguito.

```
$ su [Invio]
# mkdir /usr/share/texmf/fonts/source/hierotex [Invio]
# mkdir /usr/share/texmf/fonts/source/hierotex/mf [Invio]
# mkdir /usr/share/texmf/fonts/source/hierotex/auxmf [Invio]
# mkdir /usr/share/texmf/fonts/tfm/hierotex [Invio]
# mkdir /usr/share/texmf/tex/latex/hierotex [Invio]
# cd HieroTeX [Invio]
# cp Fonts/mf/* /usr/share/texmf/fonts/source/hierotex/mf [Invio]
```

```
# cp Fonts/auxmf/* /usr/share/texmf/fonts/source/hierotex/auxmf
[Invio]

# cp Fonts/font/*.tfm /usr/share/texmf/fonts/tfm/hierotex
[Invio]

# cp TEX/*.sty /usr/share/texmf/tex/latex/hierotex [Invio]

# cp TEX/*.fd /usr/share/texmf/tex/latex/hierotex [Invio]
```

Successivamente, occorre ricostruire i file ‘ls-R’ all’interno della struttura di LaTeX. Questo lo si può ottenere attraverso ‘texconfig’, selezionando la voce ‘REHASH’ dal menù principale.

```
# texconfig [Invio]
```

Durante l’installazione dei caratteri e degli stili, occorre fare attenzione ai permessi delle directory e dei file: i file devono essere leggibili a tutti, mentre le directory, oltre a questo, devono essere anche attraversabili.

Sesh

Sesh è un programma molto semplice, il cui scopo è quello di pre-elaborare un sorgente LaTeX, scritto inserendo caratteri geroglifici, ma in modo semplificato. Il risultato è un file LaTeX corretto, che però sarebbe più difficile da scrivere.

Questo programma è indispensabile per lavorare bene con HieroTeX, per cui è necessario procedere alla sua compilazione. Nella documentazione originale, si indica la necessità di mettere mano al file ‘HieroTeX/variable.mk’; tuttavia, per la compilazione di Sesh, ciò non dovrebbe essere necessario. Per la compilazione si entra nella directory contenente i sorgenti.

```
$ cd HieroTeX/Seshnesu [Invio]
```

```
$ make configure [Invio]
```

```
$ make sesh [Invio]
```

Se si avvia ‘make’ senza argomenti, si ottiene semplicemente un promemoria delle opzioni disponibili.

La compilazione genera il file eseguibile ‘sesh’, che può essere collocato dove si ritiene più opportuno, purché da lì possa essere utilizzato.

## Utilizzare HieroTeX

Per poter scrivere dei simboli geroglifici attraverso HieroTeX, è necessario importare uno stile di questo sistema e utilizzare i comandi relativi. Prima di analizzare la sintassi e il comportamento dei comandi specifici di HieroTeX, è opportuno iniziare con un esempio banale, in modo da verificarne il funzionamento.

```
\documentclass{report}
\usepackage{hierLtx}
\begin{document}
\begin{center}
\hieroglyph{F/35} = nfr
\end{center}
\end{document}
```

Supponendo che il file si chiami ‘prova.tex’, la sua composizione avviene nel modo solito:

```
$ latex prova.tex [Invio]
```

```
$ dvips -o prova.ps prova.dvi [Invio]
```

La stessa cosa potrebbe essere ottenuta con un esempio leggermente differente:

```
\documentclass{report}
\usepackage{hiero}
\begin{document}
\begin{center}
\begin{hieroglyph}
F35
\end{hieroglyph}
= nfr
\end{center}
\end{document}
```

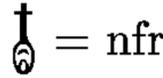
In tal caso, prima di dare in pasto questo file a LaTeX, occorre filtrarlo attraverso Sesh:

```
$ cat prova.tex | sesh > prova-1.tex [Invio]
```

```
$ latex prova-1.tex [Invio]
```

```
$ dvips -o prova.ps prova-1.dvi [Invio]
```

Figura u95.3. Il risultato ingrandito della composizione dei due esempi introduttivi.



Due modi di usare HieroTeX

A seconda delle esigenze che si hanno, si può usare HieroTeX in due modi: incorporando lo stile ‘hierLtx’ o lo stile ‘hiero’. Nel primo caso, per rappresentare i caratteri geroglifici si può usare solo il comando ‘\hieroglyphe{...}’, mentre nel secondo si usa un ambiente: ‘\begin{hieroglyph}... \end{hieroglyph}’. Tuttavia, a seconda della situazione cambia il modo in cui i simboli geroglifici vanno annotati.

In generale, con il comando ‘\hieroglyphe{...}’ si possono indicare i simboli nella forma ‘lettera / numero’, per cui,

```
\hieroglyphe{F/35}
```

corrisponde al simbolo già mostrato nella figura u95.3. Al contrario, per fare la stessa cosa nell’altro modo, bisognerebbe scrivere:

```
\begin{hieroglyph}{\leavevmode \Runh{\Aca F/35/}}\end{hieroglyph}
```

Tuttavia, disponendo dell’aiuto di Sesh, è sufficiente scrivere invece la sigla del geroglifico, nella forma ‘lettera numero’ (senza la barra):

```
\begin{hieroglyph}F35\end{hieroglyph}
```

In generale, può essere conveniente utilizzare il primo metodo solo per scrivere poche cose, in modo tale da non dipendere da Sesh per annotare uno o due simboli; ma per fare qualcosa di più, è molto meglio scegliere il secondo stile utilizzando Sesh prima della composizione.

I simboli geroglifici devono poter essere raggruppati assieme stabilendo anche la sovrapposizione eventuale. Per entrambi gli stili di scrittura si possono usare il trattino singolo (‘-’) e i due punti (‘:’), per ottenere rispettivamente la separazione orizzontale e la separazione verticale. Si osservino i due esempi seguenti che generano lo stesso risultato:

```
\hieroglyphe{M/17-X/1:N/35:N/5}
```

```
\begin{hieroglyph}M17-X1:N35:N5\end{hieroglyph}
```

Figura u95.9. ‘M17-X1:N35:N5’.



Nel caso particolare del comando ‘\hieroglyphe{...}’, si possono raggruppare più segni tra parentesi graffe; volendo scrivere in modo più preciso quanto è già stato mostrato, si potrebbero riunire i tre simboli finali:

```
\hieroglyphe{M/17-{X/1:N/35:N/5}}
```

L’ambiente ‘hieroglyph’ offre di più e questo viene descritto nella prossima sezione.

Scrittura normale

Per poter scrivere in maniera «decente» un testo con simboli geroglifici, occorre utilizzare la seconda modalità, quella che si avvale dell’aiuto di Sesh. A differenza del primo modo, i simboli possono essere indicati attraverso la sigla corrispondente, senza barre di separazione, oppure attraverso la loro traslitterazione, ammesso che esista. La codifica utilizzata deriva dal documento *Inventaire des signes hieroglyphiques en vue de leur saisie informatique*, citato alla

fine del capitolo e noto anche come *manuel de codage*, benché non sia perfettamente aderente a quel documento. A partire dalla figura u95.26 vengono elencati i codici disponibili con HieroTeX; tuttavia, dal momento che la qualità di queste immagini non è molto buona, conviene eventualmente fare riferimento alla tabella relativa contenuta nel documento *A LaTeXperiment of hieroglyphic typesetting*, sempre citato alla fine del capitolo.

I simboli, indicati attraverso la sigla standard, oppure la loro traslitterazione, possono essere separati nel modo già visto, attraverso il trattino e i due punti ('-', ':'), mentre il raggruppamento si fa attraverso l'uso delle parentesi tonde. Ma in questo ambiente sono possibili anche altri effetti, riepilogati in parte nella tabella u95.11. Inoltre, è possibile anche la scrittura incolonnata. Prima di illustrare in che modo è possibile ottenere l'incolonnamento, vengono mostrati alcuni esempi comuni, escluso il caso del raggruppamento che è già stato presentato.

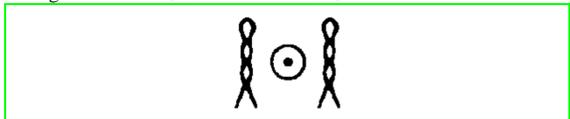
Tabella u95.11. Alcuni dei simboli speciali per la scrittura.

Codice	Risultato
-	Separa orizzontalmente.
:	Separa verticalmente.
(...)	Raggruppa.
*	Separa allo stesso livello.
#	Sovrascrive.
-=	Conclusione grammaticale.
:=	Conclusione grammaticale.
..	Spazio.
.	Mezzo spazio.
\	Ruota orizzontalmente il simbolo che lo precede.
\sn	Riduce la dimensione del simbolo che lo precede di <i>n</i> volte.
<...->	Delimita all'interno di un cartiglio.
<S...->	Delimita all'interno di un «serekh».
<Sb...->	Delimita mostrando solo l'inizio di un serekh.
<Sm...->	Delimita mostrando solo la parte centrale di un serekh.
<Se...->	Delimita mostrando solo la parte finale di un serekh.
<H...->	Delimita all'interno di un segno «hwt».
-#-...-#-	Ombreggiatura dei simboli contenuti.
+l...+s	Delimita del testo normale (LaTeX).
...-	Pone il testo normale elevato all'esponente.
\!	Avvicinamento tra i simboli.
-!	
:!	Chiude una colonna.

• Spaziatura

```
\begin{hieroglyph}
(V28-. :N5:.-V28)
\end{hieroglyph}
```

Figura u95.13. '(V28-. :N5:.-V28)'



La figura u95.13 mostra il risultato della composizione. Si osservi l'uso del punto singolo, come richiesta esplicita di un piccolo spazio, prima e dopo il simbolo N5. Senza questa spaziatura, il simbolo apparirebbe troppo basso.

• Rotazione orizzontale

```
\begin{hieroglyph}
(A1-A1\ )
\end{hieroglyph}
```

figura u95.15. '(A1-A1\ )'



La figura u95.15 mostra il risultato della composizione. L'inversione del secondo simbolo è stato ottenuto aggiungendo in coda una barra obliqua inversa ('\').

• Cartiglio

```
\begin{hieroglyph}
<-(M17-X1:N35:N5)-(G25-\!\!Aa1:.) :N35->
```

Figura u95.17. '<-(M17-X1:N35:N5)-(G25-\!\!Aa1:.) :N35->'



La figura u95.17 mostra il risultato della composizione. Il trattino utilizzato all'interno dei simboli '<' e '>' serve solo a evitare ambiguità con altri comandi particolari, ovvero con altri tipi di cornici diverse dal cartiglio.

• Avvicinamento

```
\begin{hieroglyph}
(G39-\!\!N5:.)
\end{hieroglyph}
```

Figura u95.19. '(G39-\!\!N5:.)'



```
\begin{hieroglyph}
(I10:\!\!X1:N17)
\end{hieroglyph}
```

Figura u95.21. '(I10:\!\!X1:N17)'



Le figure u95.19 e u95.21 mostrano rispettivamente i due esempi, dove nel primo caso c'è un avvicinamento di simboli in modo orizzontale, mentre nel secondo si ha un avvicinamento in modo verticale.

Vale la pena di annotare che uno o più spazi rappresentano la fine di una parola. Gli spazi vanno messi prima dei simboli di separazione (il trattino e i due punti) e questo, tra le altre cose, facilita l'incolonnamento del testo nel sorgente LaTeX.

Per incolonnare i simboli geroglifici, si utilizza il comando seguente,

```
\EnColonne[dimensione \Htm]{...}
```

dove all'interno delle parentesi graffe va dichiarato l'ambiente 'hieroglyph'. Viene mostrato un esempio abbastanza complesso, tratto dalla documentazione di HieroTeX. Viene abbinato lo stesso testo, prima in forma orizzontale, poi in forma verticale:

```
\begin{hieroglyph}
G5 <S B1 D40 xa m R19+(t:niwt) > nby wH swt-i-i (ra:Z1)mi m p:t:pt
sxm+G8+(F9:F9) (Dar:xa)*23 (sw:t)*(bit:t) <ra-mm-xpr> zA*\!\!{\!(ra:.)}
<G26-ms+mfr-xpr> O10 nb:t M:F (KA:t)*(N33:N33:N33) mr*i+1
\end{hieroglyph}
```



Figura u95.28. Codifica usata da HieroTeX.

	D61, sAH		D62		D63
	E1		E2		E3
	E4		E5		E6, zzm
	E7		E8		E8A
	E9		E10		E11
	E12		E13		E14
	E15		E16		E17, zAb
	E18		E19		E20
	E21		E22, mAi		E23, rw, l
	E24, Aby		E25		E26
	E27		E28		E29
	E30		E31		E32
	E33		E34, wn		F1
	F2		F3		F4, HAt
	F5, SsA		F6		F7
	F8		F9		F10
	F11		F12, wsr		F13, wp
	F14		F15		F16, db
	F17		F18, Hw, bH		F19
	F20, ns		F21, idn, msDr, sDm, DrD		F22, pH, kFA
	F23, xpS		F24		F25, wHm
	F26, Xn		F27		F28
	F29, sti		F30, Sd		F31, ms
	X, F32		F33, sd		F34, ib

Figura u95.30. Codifica usata da HieroTeX.

	G51		G52		G53
	G54, snD		H1		H2, wSm
	H3, pAq		H4		H5
	H6, Sw		H6A		H7
	H8		I1, aSA		I2, Styw
	I3, mzH		I4, sbk		I5, sAq
	I5A		I6, km		I8, Hfn
	f, I9		D, I10		I11, DD
	I12		I13		I14
	I15		K1, in		K2
	K3, ad		K4, XA		K5, bz
	K6, nSmt		K7		L1, xpr
	L2, bit		L3		L4
	L5		L6		L7, srqt
	M1, iAm		M2, Hn		M3, xt
	M4, rnp		M5		M6, tr
	M7		M8, SA		M9, zSu
	M10		M11, wdn		M12, xA
	M13, wAD		M14		M15
	M16, HA		i, M17		M18, ii
	M19		M20, sxct		M21, sm
	M22		M23, sw		M24, rsw
	M25		M26, Sma		M27
	M28		M29, nDm		M30, bnr

Figura u95.29. Codifica usata da HieroTeX.

	F35, nfr		F36, zmA		F37
	F37B		F38		F39, imAx
	F40, Aw		F41		F42, spr
	F43		F44, iwa, isw		F45
	F46, pXr, qAb		F47		F48
	F49		F50		F51
	F52		A, G1, A		G2, AA
	G3		G4, tyw		G5
	G6		G7		G7A
	G7AA		G8		G9
	G10		G11		G12
	G13		G14, mwt		G15
	G16, nbty		m, G17		G18, mm
	G19		G20		G21, nH
	G22, Db		G23, rxyt		G24
	G25, Ax		G26		G26A
	G27, dSr		G28, gm		G29, bA
	G30, bAw		G31		G32, baHi
	G33		G34		G35, aq
	G36, wr		G37		G38, gb
	G39, zA		G40, pA		G41, xn, pA'
	G42, wSA		w, G43		G44, ww
	G45		G46, mAw		G47, TA
	G48		G49		G50

Figura u95.31. Codifica usata da HieroTeX.

	M31		M32		M33
	M34, bdt		M35		M36, Dr
	M37		M38		M39
	M40, iz		M41		M42
	M43		M44		N1, pt
	N2		N3		N4, iAdt, idt
	N5, ra, zw, hrw		N6		N7
	N8, Hnmmt		N9, pzD		N10
	N11, Abd, iaH		N12		N13
	N14, dwA, sbA		N15, dwAt		N16, tA
	N17		N18, iw		N19
	N20, wDb, idb		N21		N22
	N23		N24, spAt		N25, xAst
	N26, Dw		N27, Axt		N28, xa
	q, N29		N30, iAt		N31
	N32		N33		N33A
	N34		n, N35		N35A, mw
	N36		S, N37		N38
	N39		N40, Sm		N41, id, N42
	O1, pr		O2		O3
	h, O4		O5		O6, Hwt
	O7		O8		O9
	O10		O11, aH		O12
	O13		O14		O15, wsxt

Figura u95.32. Codifica usata da HieroTeX.

	O16		O17		O18, kAr
	O19		O20		O21
	O22, zH		O23		O24
	O25, txn		O26		O27
	O28, iwn		O29, aA		aAv, O29v
	O30, zcnt		O31		O32
	O33		z, O34		O35, zb
	O36, imb		O37		O38
	O39, inr		O40		O41
	O42, Ssp		O43		O44
	O45, ipt		O46		O47, nxn
	O48		O49, niwt		O50, zp
	O51, Sswt		P1		P1A
	P2		P3		P4, wHa
	P5, TAw, nfw		P6, aHa		P7
	P8, xrw		P9		P10
	P11		Q1, st		Q2, wz
	p, Q3, p		Q4		Q5
	Q6, qrsW, qrs		Q7		R1, xAwt, xAt
	R2		R3		R4, Htp
	R5, kAp, kp		R6		R7, snTr
	R8, nTr		R9, bd		R10
	R11, dd, Dd		R12		R13
	R14, imnt		R15, iAb		R16, wx

Figura u95.34. Codifica usata da HieroTeX.

	T15		T16		T17, wrrt
	T18, Sms		T19, qs		T20
	T21, wa		T22, sn		T23
	T24, iH		T25, DbA		T26
	T27		T28, Xr		T29, nmt
	T30		T31, sSm		T32
	T33		T34, nm		T35
	U1, mA		U2		U3
	U4		U5		U6, mr
	U7		U8		U9
	U10, it		U11, HqAt		U12
	U13, hb, Sna		U14		U15, tm
	U16, biA		U17, grg		U18
	U19		U20		U21, stp
	U22, mnx		U23, Ab		U24, Hmt
	U25		U26, wbA		U27
	U28, DA		U29		U30
	U31, rtH		U32, zmn		U33, ti
	U34, xsf		U35		U36, Hm
	U37		U38, mxAt		U39
	U40		U41		V1, St, Snt, 100
	V2, sTA		V3, sTAW		V4, wA
	V5, snT		V6, Ss		V7, Sn
	V8		V9		V10

Figura u95.33. Codifica usata da HieroTeX.

	R17		R18		R19
	R20		R21		R22, xm
	R23		R24		R25
	S1, HDt		S2		N, S3, dSrt
	S4		S5		S6, sxnty
	S7, xprS		S8, Atf		S9, Swty
	S10, mDH		S11, wsx		S12, nbw
	S13		S14		S14A
	S15, tHn, THn, S16		S17		S17A
	S18, mmit		S19, sDAw		S20, xtm
	S21		S22, sT		S23, dmD
	S24, Tz		S25		S26, Sndyt
	S27, mnx		S28		s, S29
	S30, sf		S31		S32, siA
	S33, Tb		S34, aux		S35, Swt
	S36		S37, xw		S38, HqA
	S39, awt		S40, wAs		S41, Dam
	S42, abA, sxm, xrp		S43, md		S44, Ams
	S45, nxxw		T1		T2
	T3, HD		T4		T5
	T6, HDD		T7		T7A
	T8		T8A		T9, pd
	T9A		T10, pD		T11, zin, zwn, sXr
	T12, Ai, Ar, rwd, rWD		T13, rs		T14, qmA

Figura u95.35. Codifica usata da HieroTeX.

	V11		V12, arq		T, V13, T
	V14		V15, iTi		V16
	V17		V18		V19, mDt, XAr, TmA
	V20, 10, mD		V21		V22, mH
	V23		V24, wD		V25
	V26, aD		V27		H, V28
	V29, wAH, sk		V30, nb		k, V31
	V31A, k'		V32, msn		V33, sSr
	V34		V35		V36
	V37, idr		V38		V39
	W1		W2, bAs		W3, Hb
	W4		W5		W6
	W7		W8		W9, Xnm
	W10, iab		W10A		g, W11, nst
	W12		W13		W14, Hz
	W17, xnt		W18		W19, mi
	W20		W21		W22, Hnqt
	W23		W24, nw		W25, ini
	t, X1		X2		X3
	X4		X5		X6
	X7		X8, rdi, di		Y1, mDA
	Y1v		Y2		Y3, zS, mmhd
	Y4		Y5, mn		Y6, ibA
	Y7		Y8, zSS		Z1

Figura u95.36. Codifica usata da HieroTeX.

⋮	Z2	⋮	Z3	⋮	Z3A
∞	Z4, y	∞	Z5	∞	Z6
∞	W, Z7	∞	Z8	×	Z9
×	Z10	+	Z11, imi, wmm	‘	Z98A
∞	spd, Z99A	⊙	x, Aa1	∞	Aa2
∞	Aa3	∞	Aa4	∞	Aa5, Hp
∞	Aa6	∞	Aa7	∞	Aa8, qn
∞	Aa9	∞	Aa10	∞	Aa11, mAa
∞	Aa12	∞	M, Aa13, im, gs	∞	Aa14
∞	Aa15	∞	Aa16	∞	Aa17, sA
∞	Aa18	∞	Aa19	∞	Aa20, apr
∞	Aa21, wDa	∞	Aa22	∞	Aa23
∞	Aa24	∞	Aa25	∞	Aa26
∞	Aa27, nD	∞	Aa28, qd	∞	Aa29
∞	Aa30, Xkr	∞	Aa31	∞	Aa32

## Riferimenti

- Serge Rosmorduc, *A Short Introduction to Hieroglyphs*  
<http://www.iut.univ-paris8.fr/~rosmod/Intro/Intro.html>
- Jan Buurman, Nicolas Grimal, Michael Hainsworth, Jochen Hallof, Dirk Van Der Plas, *Inventaire des signes hieroglyphiques en vue de leur saisie informatique*, Mémoires de l'Académie des Inscriptions et Belle Lettres, Institut de France, Paris, 1988

## Trasformazione in altri formati

DLH: trasforma LaTeX in HTML .....	721
Help2man: genera una pagina di manuale dalle informazioni fornite dal programma .....	722
Pstotext: estrae il testo da un file PostScript o PDF .....	723
Mswordview .....	723
Catdoc .....	723
Antiword .....	724
xLHTML .....	725

Spesso ci si trova di fronte alla necessità o all'utilità di trasformare un documento scritto in un certo modo, per esempio in LaTeX, in qualcosa di diverso, per esempio in HTML. In generale, queste cose andrebbero pianificate prima, per decidere lo stile del documento in base alle forme in cui questo deve poi concretizzarsi. Meglio ancora sarebbe l'utilizzo di strumenti appositi, di solito SGML/XML, pensati in anticipo per la produzione di documentazione in formati differenti.

Questo capitolo serve a raccogliere la descrizione di strumenti che possono aiutare a trasformare un documento realizzato con sistemi di composizione tradizionale, pensati principalmente per la stampa su carta, e viceversa.

Non ci si possono fare illusioni: gli strumenti di questo tipo non funzionano sempre, ma solo quando le caratteristiche del sorgente lo consentono.

### DLH: trasforma LaTeX in HTML

DLH<sup>1</sup> è uno strumento relativamente semplice per la conversione di sorgenti LaTeX in HTML. La trasformazione avviene con successo solo quando si tratta di un sorgente LaTeX in cui non si usano ambienti matematici e soprattutto non si usano comandi particolarmente sofisticati (ciò inteso dal punto di vista di DLH).

DLH utilizza un insieme personalizzato di stili LaTeX, collocato normalmente nella directory `'/usr/share/dlh/inputs/dlh/'`. Si tratta dei soliti `'article.sty'`, `'epsfig.sty'` e altri, ma il contenuto di questi file è ridotto rispetto a quelli equivalenti di LaTeX. Se nel sorgente LaTeX si utilizzano altri stili particolari occorrerebbe creare un file corrispondente anche in questa directory, cercando di adattarlo a DLH (cosa che potrebbe risultare difficile, dal momento che bisogna ragionare in termini di TeX limitato secondo le possibilità di DLH).

Il programma eseguibile è `'dlh'` che accetta l'indicazione di alcune opzioni e in particolare un elenco di file LaTeX:

```
dlh [opzioni] file_latex...
```

In corrispondenza dei file indicati come argomento vengono create altrettante directory contenenti una serie di file HTML che rappresentano il risultato della trasformazione (a partire da `'index.html'` che normalmente è un collegamento simbolico al primo di questi file).

DLH utilizza una serie di icone per rappresentare i pulsanti per lo scorrimento del documento secondo la sua struttura. I file di queste icone si trovano normalmente nella directory `'/usr/share/dlh/icons/'` e andrebbero copiati nella directory `'../icons/'`, rispetto a quella in cui si trovano i file HTML.

Tabella u96.1. Alcune opzioni.

Opzione	Descrizione
-f --force	Questa opzione serve a creare tutti i file che compongono il documento, in particolare le immagini. Ciò può creare un rallentamento nel funzionamento di DLH, ma in generale serve a garantire un risultato più sicuro.
-i <i>uri</i> --icon-dir= <i>uri</i>	Permette di definire esplicitamente la collocazione dei file che rappresentano le icone utilizzate da DLH per rappresentare i pulsanti per lo scorrimento del documento.

Segue la descrizione di alcuni esempi.

- \$ `dlh prova.tex [Invio]`

Crea la directory `./prova/` e al suo interno inserisce una serie di file HTML che riproducono il documento `prova.tex`. In questo caso, i file HTML fanno uso delle icone che si trovano nella directory `./icons/`, relativa al nodo di rete in cui si trovano.

- \$ `dlh -f prova.tex [Invio]`

Come nell'esempio precedente, ma viene forzata la creazione di tutti i file, nel caso ce ne fosse bisogno.

- \$ `dlh -i icone prova.tex [Invio]`

Come nel primo esempio, con la differenza che i file delle icone devono trovarsi nella directory `./prova/icone/`.

Help2man: genera una pagina di manuale dalle informazioni fornite dal programma

Help2man <sup>2</sup> è un programma in grado di generare una pagina di manuale a partire dalle informazioni che restituisce un altro programma attraverso le opzioni `--help` e `--version`.

Help2man è predisposto principalmente per gestire convenientemente il risultato generato da un programma che segue le convenzioni GNU (ovvero della Free Software Foundation).

```
help2man [opzioni] programma_eseguibile
```

Lo schema sintattico permette di vedere che si tratta dell'eseguibile `help2man`, che oltre alle opzioni eventuali richiede l'indicazione di un programma da avviare con le opzioni `--help` e `--version` per ottenere le informazioni necessarie. In modo predefinito, il risultato viene emesso attraverso lo standard output.

Tabella u96.2. Alcune opzioni.

Opzione	Descrizione
-o <i>file</i> --output= <i>file</i>	Permette di definire il nome del file da generare, evitando così di emettere il risultato attraverso lo standard output.
-s <i>n_sezione</i> --section= <i>n_sezione</i>	Permette di specificare il numero della sezione della pagina di manuale.

Segue la descrizione di alcuni esempi.

- \$ `help2man ls > ls.1 [Invio]`

Genera il file `ls.1`, contenente la pagina di manuale di `ls`.

- \$ `help2man -o ls.1 ls [Invio]`

Esattamente come nell'esempio precedente, utilizzando esplicitamente l'opzione `-o`.

Pstotext: estrae il testo da un file PostScript o PDF

Pstotext <sup>3</sup> è un programma molto semplice per l'estrazione del testo contenuto all'interno di un file PostScript o PDF, per mezzo di Ghostscript.

```
pstotext [opzioni] file
```

Tutto il lavoro viene svolto dall'eseguibile `pstotext`. Il risultato dell'elaborazione viene emesso attraverso lo standard output, a meno che sia stato stabilito diversamente con le opzioni.

Tabella u96.3. Alcune opzioni.

Opzione	Descrizione
-cork	Specifica che il file PostScript utilizza la codifica «cork», ovvero ciò che si ottiene da Dvips quando questo converte file DVI generati da TeX con la codifica T1.
-landscape -landscapeOther	Queste due opzioni indicano che il testo è ruotato a 90 gradi in un senso, oppure nell'altro.
-portrait	In questo caso si intende che il testo scorre come di consueto, su un foglio orientato in modo verticale.
-output <i>file</i>	Consente di indicare il file di testo da generare, senza bisogno di ridirigere lo standard output.

Mswordview

Mswordview <sup>4</sup> è un programma il cui scopo è quello di convertire file di MS-Word in HTML. La conversione non può essere perfetta, ma il progetto è condotto con impegno e i risultati che dà questo programma sono buoni.

L'eseguibile di questo programma corrisponde a `mswordview` e la sintassi per il suo utilizzo si può schematizzare secondo il modello seguente:

```
mswordview [opzioni] file_doc
```

Mswordview è in grado di convertire solo un file alla volta, precisamente quello che viene indicato alla fine degli argomenti. Se non viene richiesto qualcosa di particolare attraverso le opzioni, Mswordview tenta di creare un file con lo stesso nome di quello che viene convertito, con l'aggiunta dell'estensione `.html`. Inoltre, se il file contiene delle immagini incorporate, queste vengono trasferite su file esterni.

Tabella u96.4. Alcune opzioni.

Opzione	Descrizione
-o <i>file_html</i> --outputfile <i>file_html</i>	Permette di indicare esplicitamente il file HTML che si vuole generare.
-g <i>file_errori</i> --errorfile <i>file_errori</i>	Permette di annotare gli errori incontrati durante la conversione nel file indicato.

Catdoc

Catdoc <sup>5</sup> è un programma molto semplice, che si sostituisce idealmente a `cat` quando si tratta di visualizzare il contenuto di file scritti in formato MS-Word. Il suo funzionamento è intuitivo e in generale non servono opzioni: il file indicato come argomento, o fornito attraverso lo standard input, viene emesso dallo standard output dopo una conversione in formato testo. Se il file originale contiene in realtà solo testo puro, non avviene alcuna conversione.

```
catdoc [opzioni] file_doc
catdoc [opzioni] < file_doc
```

Tabella u96.5. Alcune opzioni.

Opzione	Descrizione
-b	Cerca di elaborare anche file MS-Word che apparentemente non lo sono, a causa di una firma iniziale errata.
-mn	Specifica il margine destro del testo ottenuto. Il margine predefinito è a colonna 72. Si osservi che l'opzione '-m0' equivale a '-w'.
-w	Specifica il margine destro del testo ottenuto di lunghezza indefinita, in modo da ottenere che i paragrafi occupino una riga intera.
-v	Genera alcune informazioni diagnostiche prima del testo trasformato.

Per quanto semplice possa essere questo programma, è prevista una configurazione, composta dal file `/etc/catdocrc` per il sistema e dai file `~/catdocrc` per gli utenti. Senza entrare nel dettaglio delle direttive di configurazione, è il caso di descrivere quella che rappresenta l'impostazione comune:

```
charset_path=/usr/lib/catdoc
map_path=/usr/lib/catdoc
source_charset=cp1252
target_charset=UTF-8
unknown_char='?'
```

Come si può intuire, le direttive `'charset_path'` e `'map_path'` servono a indicare la collocazione di file utilizzati da Catdoc per la conversione. La direttiva `'source_charset'` permette di stabilire la codifica predefinita del file sorgente, quando questo non appare utilizzare la UTF-16. La direttiva `'target_charset'` permette di definire la codifica da usare per il testo generato; come si vede nell'esempio viene usata la codifica UTF-8. Infine, è possibile stabilire in che modo mostrare i caratteri che non possono essere rappresentati, attraverso la direttiva `'unknown_char'`, che in questo caso usa il punto interrogativo.

Segue la descrizione di alcuni esempi.

- `$ catdoc pippo.doc | less [Invio]`

Visualizza il contenuto del file `'pippo.doc'`, con l'aiuto di `'less'` per scorrerlo.

- `$ catdoc pippo.doc > pippo.txt [Invio]`

Genera il file `'pippo.txt'` a partire da `'pippo.doc'`.

## Antiword

Antiword <sup>6</sup> è un programma molto semplice per convertire file dal formato MS-Word in testo puro e semplice, oppure in PostScript, estrapolando anche le immagini. Il suo funzionamento è intuitivo e in generale non servono opzioni: il file indicato come argomento, viene emesso attraverso lo standard output dopo la conversione.

```
antiword [opzioni] file_doc...
```

Tabella u96.7. Alcune opzioni.

Opzione	Descrizione
-t	Genera una conversione in formato testo puro e semplice. L'uso di questa opzione è implicito.
-w <i>n_colonne</i>	Permette di specificare, nell'ambito di una conversione in formato testo, l'ampiezza del testo in caratteri. Se si utilizza il valore zero, si ottiene ogni paragrafo in una sola riga.
-m <i>file_mappa</i>	Consente di indicare la codifica del file di testo che si vuole ottenere.

Opzione	Descrizione
-p <i>dimensioni_carta</i>	L'utilizzo di questa opzione richiede implicitamente la conversione in formato PostScript, mentre in condizioni normali si ottiene un testo puro e semplice. L'argomento dell'opzione stabilisce la dimensione della carta e può trattarsi delle parole chiave seguenti, con il significato intuitivo che hanno: <code>'10x14'</code> , <code>'a3'</code> , <code>'a4'</code> , <code>'a5'</code> , <code>'b4'</code> , <code>'b5'</code> , <code>'executive'</code> , <code>'folio'</code> , <code>'legal'</code> , <code>'letter'</code> , <code>'note'</code> , <code>'note'</code> , <code>'quarto'</code> , <code>'statement'</code> , <code>'tabloid'</code> .
-L	Nell'ambito di una conversione in PostScript, indica un orientamento orizzontale del foglio.
-i <i>livello_di_visualizzazione_immagini</i>	Consente di specificare cosa fare delle immagini che fossero eventualmente contenute nel file di partenza. L'argomento è un numero.
-i 0	Genera un file compatibile con Ghostscript, ma non adatto a stampanti PostScript comuni. Tuttavia, in condizioni normali, se si arriva alla stampa, si passa generalmente per Ghostscript, per cui questo valore è quello che può essere adatto.
-i 1	Non estrapola le immagini.
-i 2	PostScript livello 2.
-i 3	PostScript livello 3.
-s	Include anche il testo nascosto, indicato come tale nel file originale.

Segue la descrizione di alcuni esempi.

- `$ antiword pippo.doc | less [Invio]`

Visualizza il contenuto del file `'pippo.doc'`, con l'aiuto di `'less'` per scorrerlo.

- `$ antiword -p a4 pippo.doc > pippo.ps [Invio]`

Genera il file `'pippo.ps'` (PostScript, A4) a partire da `'pippo.doc'`.

## xIHTML

xIHTML <sup>7</sup> è un programma per convertire file dal formato MS-Excel in HTML, come suggerisce il nome, oppure in testo puro. Se non si usano le opzioni, si ottiene un file HTML, contenente una tabella con ciò che appare nel foglio elettronico indicato nella riga di comando, emesso attraverso lo standard output:

```
xlhtml [opzioni] file_xls > file
```

Tabella u96.8. Alcune opzioni.

Opzione	Descrizione
-fw	In condizioni normali, le celle che contengono delle espressioni vengono valutate e ne viene mostrato solo il risultato, con una nota sulla possibilità che il valore mostrato non sia preciso. Per fare sparire questa nota si usa l'opzione <code>'-fw'</code> .
-c	Fa in modo che la tabella contenente i dati del foglio elettronico appaia centrata nel corpo della pagina HTML.
-asc	Genera un risultato in formato testo puro, se però si abbina anche una delle opzioni che iniziano per <code>'-x'</code> .

Opzione	Descrizione
-csv	Genera un risultato in formato testo, dove i campi sono delimitati da apici doppi e separati da una virgola. Anche in questo caso, l'opzione funziona solo in abbinamento con una delle opzioni che iniziano per '-x'.
-xp:n	Converte solo la pagina <i>n</i> , contando a cominciare da zero.
-xc:m-n	Converte solo le colonne da <i>m</i> a <i>n</i> , contando a cominciare da zero.
-xr:m-n	Converte solo le colonne da <i>m</i> a <i>n</i> , contando a cominciare da zero.

Per comprendere le possibilità di xHTML viene mostrato un solo esempio, di un foglio elettronico realizzato con Gnumeric, salvando in formato XLS. Due figure mostrano il contenuto del foglio, sia nel suo aspetto finale, sia nel contenuto effettivo delle celle.

Figura u96.9. Il foglio elettronico di esempio, nel suo aspetto finale.

	A	B	C	D	E
1	Straordinario				
2	Data	dalle ore	alle ore	durata	
3	15/01/2005	19:30	21:30	2:00	
4	16/01/2005	19:30	21:30	2:00	
5	17/01/2005	19:30	21:30	2:00	
6	Totale straordinario:			6:00	
7					
8					

Figura u96.10. Il foglio elettronico di esempio con le espressioni contenute nelle celle.

	A	B	C	D	E
1	Straordinario				
2	Data	dalle ore	alle ore	durata	
3	=date(2005;1;15)	=time(19;30;0)	=time(21;30;0)	=C3-B3	
4	=date(2005;1;16)	=time(19;30;0)	=time(21;30;0)	=C4-B4	
5	=date(2005;1;17)	=time(19;30;0)	=time(21;30;0)	=C5-B5	
6	Totale straordinario:			=sum(D3:D5)	
7					

Supponendo che il file si chiami 'esempio.xls', si può procedere con il comando seguente per generare il file 'esempio.html':

```
$ xhtml -fw esempio.xls > esempio.html [Invio]
```

Il file che si ottiene dovrebbe avere l'aspetto seguente; si osservi che le date non sono state rappresentate in modo corretto:

**Sheet1**

Straordinario			
Data	dalle ore	alle ore	durata
38367 *	19:30	21:30	2:00
38368 *	19:30	21:30	2:00
38369 *	19:30	21:30	2:00
Totale straordinario:			6:00

Spreadsheet's Author: Unknown  
 Last Updated with Excel 97  
 \* This cell's format is not supported.

Created with xhtml 0.5.1

- 1 **DLH** GNU GPL
- 2 **Help2man** GNU GPL
- 3 **Pstotext** licenza speciale
- 4 **Mswordview** GNU GPL + alcuni file con licenza speciale
- 5 **catdoc** GNU GPL
- 6 **Antiword** GNU GPL
- 7 **xHTML** GNU GPL