Lout [1] 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 composi-

zione 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.

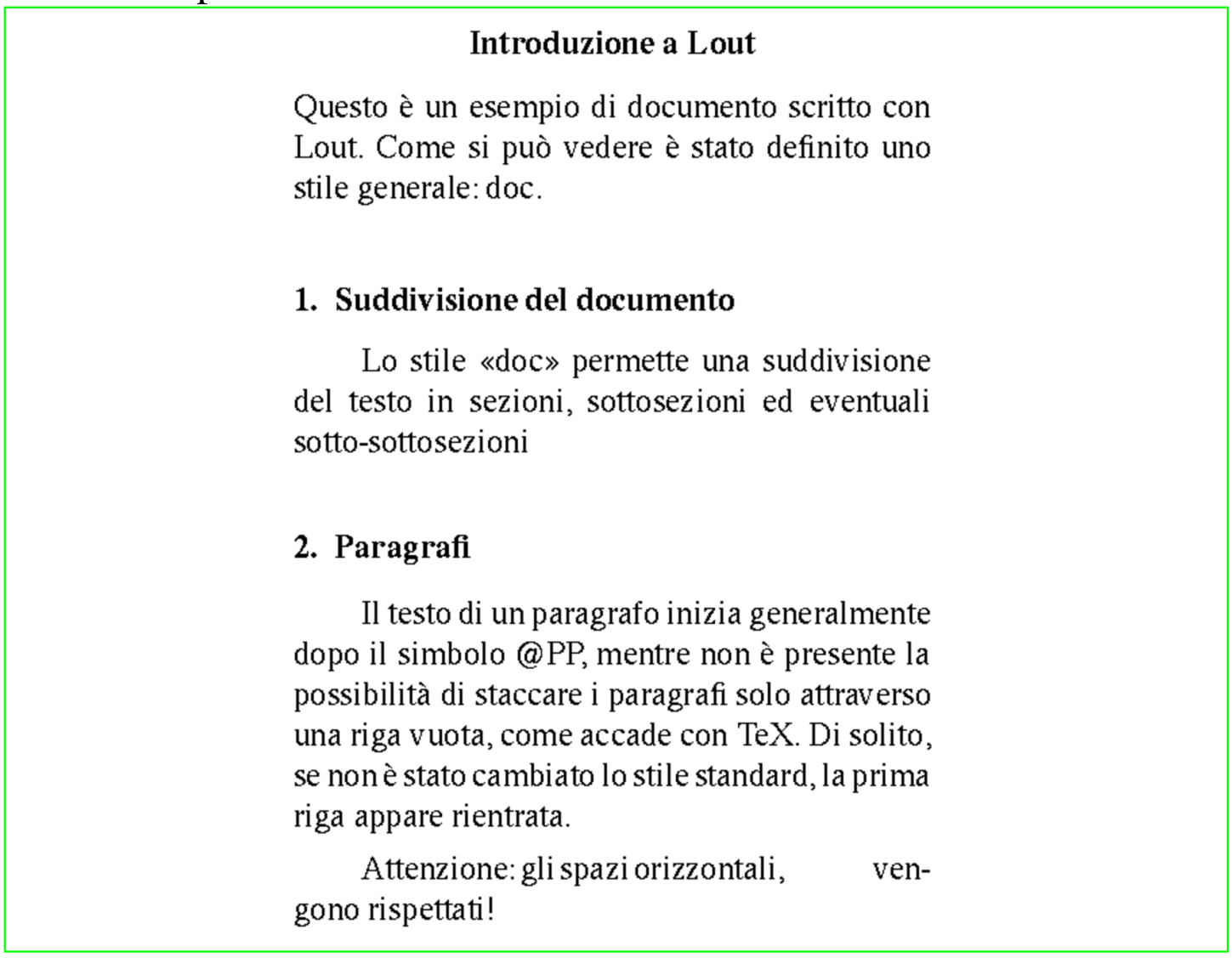
```
lout file "esempio":
    25,1: unresolved cross reference @SectionList&&357.esempio.1
    25,1: unresolved cross reference @SectionList&&357.esempio.1
    35,1: unresolved cross reference @SectionList&&357.esempio.2
    35,1: unresolved cross reference @SectionList&&357.esempio.2
```

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 intervengono 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)	" "	"	"\""
#	"#"	&	"&"
/	"/"	@	"@"
\	"\\"	^	"^"
{	"{"	|	"|"
}	"}"	~	"~"

Spazi e spaziature

«

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. Ben-

ché 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.

Opzione	Descrizione
`@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**'; suc-

cessivamente 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.

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 inserire 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**'	Quadratoni ('**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 aacute`	á	`@Char Aacute`	Á
`@Char acircumflex`	â	`@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
@Char ecircumflex	ê	@Char Ecircumflex	Ê
@Char egrave	è	@Char Egrave	È
@Char edieresis	ë	@Char Edieresis	Ë
@Char iacute	í	@Char Iacute	Í
@Char icircumflex	î	@Char Icircumflex	Î
@Char igrave	ì	@Char Igrave	Ì
@Char idieresis	ï	@Char Idieresis	Ï
@Char ntilde	ñ	@Char Ntilde	Ñ
@Char oacute	ó	@Char Oacute	Ó
@Char ocircumflex	ô	@Char Ocircumflex	Ô

`@Char ograve`	ò	`@Char Ograve`	Ò
`@Char oslash`	ø	`@Char Oslash`	Ø
`@Char otilde`	õ	`@Char Otilde`	Õ
`@Char odieresis`	ö	`@Char Odieresis`	Ö
`@Char germandbls`	ß		
`@Char uacute`	ú	`@Char Uacute`	Ú
`@Char ucircumflex`	û	`@Char Ucircumflex`	Û
`@Char ugrave`	ù	`@Char Ugrave`	Ù
`@Char udieresis`	ü	`@Char Udieresis`	Ü
`@Char yacute`	ý	`@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 graffe. 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_maiuscoletto }
```

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 rappresenta 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**’, ‘**@UCAlphaList**’, e dalla serie parallela ‘**@ParenNumberedList**’, ‘**@ParenRomanList**’, ‘**@ParenUCRomanList**’, ‘**@ParenAlphaList**’, ‘**@ParenUCAlphaList**’. 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 |
@UCAlphaList
@ListItem elemento
@ListItem elemento
    ...
@EndList
```

```
@ParenNumberedList | @ParenRomanList | @ParenUCRomanList↩
↪ | @ParenAlphaList | @ParenUCAlphaList
@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 @I { standard output } }
@F @Verbatim @Begin
                    .-----------.
                    |           |------------>  STDOUT
STDIN   ----------->| Programma |
                    |           |------------>  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 sottovoci, 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 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*

[1] **Lout** GNU GPL