

Enrico Gregorio

L<sup>A</sup>T<sub>E</sub>X

Breve guida ai pacchetti  
di uso più comune

Edizione di novembre 2010

## Introduzione

Una *breve guida* non può sostituire i più ampi manuali disponibili con ogni distribuzione moderna di L<sup>A</sup>T<sub>E</sub>X. Vuole invece dare una rapida indicazione di come poter usare i principali comandi e ambienti messi a disposizione dai pacchetti descritti.

Non trattiamo qui dell'installazione; i primi sedici pacchetti che verranno analizzati *devono* essere presenti in tutte le distribuzioni fin dall'inizio. L'unico di questi che richiede un intervento dell'utente è **babel**, per caricare in memoria le regole di sillabazione delle lingue; si faccia riferimento al manuale della distribuzione per capire come si fa.

Di tutti i comandi è specificata la sintassi, cioè gli argomenti opzionali e obbligatori. Un'espressione come  $\langle abcde \rangle$  è un segnaposto per il vero argomento da dare; in tutti i casi dovrebbe essere chiaro che tipo di argomento è richiesto. Per esempio,  $\langle testo \rangle$  indica qualunque lista di caratteri che sia comprensibile da L<sup>A</sup>T<sub>E</sub>X in quel contesto; invece  $\langle dimen \rangle$  indica che va inserita una lunghezza esplicita oppure un parametro dimensionale come `\parindent` o simili.

Per ogni pacchetto sono specificate le opzioni disponibili e quali siano quelle usate se non ne vengono dichiarate esplicitamente. Una notazione come

```
\usepackage[ $\langle opzione \rangle, \dots$ ]{pippo}
```

significa che il pacchetto **pippo** può prendere più opzioni. Il carattere '`|`' fra due opzioni o comandi significa che si può scegliere fra due forme alternative oppure che si tratta di opzioni che si escludono a vicenda. Questo dovrebbe risultare chiaro dalla descrizione.

L'ordine in cui i pacchetti sono trattati è: **babel**, poi i pacchetti obbligatori in ogni installazione di L<sup>A</sup>T<sub>E</sub>X, quindi altri.

Suggerimenti o critiche sono bene accetti, possono essere indirizzati a

```
Enrico <dot> Gregorio <at> univr <dot> it
```

specificando nell'oggetto del messaggio **Commenti alla breve guida**. Un ringraziamento a quelli che hanno già segnalato errori e miglioramenti. Lo stesso indirizzo può essere usato da chiunque desideri segnalare involontarie violazioni di *copyright*.

Queste note sono tratte dalle documentazioni pubblicamente disponibili dei pacchetti qui descritti. È doveroso citarne gli autori:

- Johannes Braams per `babel`;
- Claudio Beccari per la versione italiana di `babel`;
- David Carlisle per `afterpage`, `array`, `bm`, `dcolumn`, `enumerate`, `ifthen`, `indentfirst`, `longtable`, `showkeys`, `tabularx` oltre a `graphicx` e `color` con Sebastian Rahtz e il L<sup>A</sup>T<sub>E</sub>X3 team;
- Frank Mittelbach per `array`, `bm`, `multicol`, `varioref`;
- Kresten Krab Thorup e Frank Jensen (con Chris Rowley) per `calc`;
- Kent McPherson per `layout`;
- Rainer Schöpf, Bernd Raichle e Chris Rowley per `verbatim`;
- Axel Sommerfeldt per `caption`;
- Anselm Lingnau per `float`;
- Steven Douglas Cochran per `subfig`;
- Bernd Schandl per `paralist`;
- Simon Fear per `booktabs`;
- Heiko Oberdiek per `ifpdf`;
- Javier Bezos per `enumitem`;
- Donald Arsenau per `url`;
- Peter Wilson per `tocloft`;
- Piet van Ostrum per `fancyhdr`;
- Karl Wette per `emptypage`;
- Hideo Umeki per `geometry`.

Cambiamenti rispetto alla versione 2006:

- correzione di errori di stampa;
- aggiunta della discussione su `\cleardoublepage` a margine del capitolo su `afterpage`;
- aggiunta dei capitoli su `booktabs` e `ifpdf`.

Cambiamenti rispetto alla versione 2007:

- correzione di errori di stampa e di impaginazione;
- menzione di alcuni pacchetti simili a `ifpdf`, nel capitolo dedicato a questo;
- aggiunta dei capitoli su `enumitem`, `url`, `tocloft`, `graphicx` e `color`.

Cambiamenti rispetto alla versione 2008:

- aggiunta dei capitoli su `fancyhdr`, `geometry` e `emptypage`.

## Indice

<b>1</b>	<b>babel</b>	<b>7</b>
1.1	Opzioni . . . . .	7
1.2	Comandi . . . . .	7
1.3	Comandi specifici per l'italiano . . . . .	9
<b>2</b>	<b>afterpage</b>	<b>10</b>
<b>3</b>	<b>array</b>	<b>11</b>
3.1	Specificatori . . . . .	11
3.2	Comandi . . . . .	12
<b>4</b>	<b>bm</b>	<b>14</b>
4.1	Comandi . . . . .	14
4.2	Esempi . . . . .	14
<b>5</b>	<b>calc</b>	<b>15</b>
5.1	Introduzione . . . . .	15
5.2	Operazioni . . . . .	15
5.3	Comandi . . . . .	16
<b>6</b>	<b>dcolumn</b>	<b>17</b>
6.1	Esempi . . . . .	17
6.2	Osservazione . . . . .	17
<b>7</b>	<b>enumerate</b>	<b>18</b>
7.1	Descrizione . . . . .	18
7.2	Uso . . . . .	18
<b>8</b>	<b>ifthen</b>	<b>20</b>
8.1	Comandi . . . . .	20
8.2	I test . . . . .	20
<b>9</b>	<b>indentfirst</b>	<b>22</b>
<b>10</b>	<b>layout</b>	<b>23</b>
10.1	Opzioni . . . . .	23
<b>11</b>	<b>longtable</b>	<b>25</b>
11.1	Opzioni . . . . .	25
11.2	Descrizione . . . . .	25
11.3	L'ambiente <code>longtable</code> . . . . .	26
11.4	Parametri . . . . .	26
11.5	Argomenti opzionali a <code>\begin{longtable}</code> . . . . .	26
11.6	Comandi per terminare le righe della tabella . . . . .	26
11.7	Comandi per la didascalia . . . . .	27
11.8	Comandi impiegabili all'inizio di una riga . . . . .	27

<b>12 multicol</b>	<b>29</b>
12.1 Opzioni . . . . .	29
12.2 Comandi . . . . .	29
12.3 Parametri . . . . .	30
<b>13 showkeys</b>	<b>32</b>
13.1 Opzioni . . . . .	32
<b>14 tabularx</b>	<b>33</b>
<b>15 varioref</b>	<b>34</b>
15.1 Opzioni . . . . .	34
15.2 Comandi . . . . .	34
15.3 I comandi che producono i testi variabili . . . . .	36
<b>16 verbatim</b>	<b>38</b>
16.1 Ambienti . . . . .	38
16.2 Creare nuovi ambienti di tipo <code>verbatim</code> . . . . .	38
<b>17 caption</b>	<b>39</b>
17.1 Opzioni . . . . .	39
17.2 Comandi . . . . .	42
17.3 Altri pacchetti . . . . .	43
<b>18 float</b>	<b>44</b>
18.1 Stili . . . . .	44
18.2 Comandi . . . . .	44
18.3 L'argomento di posizione <code>H</code> . . . . .	45
<b>19 subfig</b>	<b>46</b>
19.1 Opzioni . . . . .	46
19.2 Comandi . . . . .	47
<b>20 paralist</b>	<b>49</b>
20.1 Opzioni . . . . .	49
20.2 Ambienti . . . . .	50
20.3 Comandi . . . . .	51
20.4 Parametri . . . . .	52
<b>21 booktabs</b>	<b>53</b>
21.1 Comandi per i filetti orizzontali . . . . .	53
21.2 Esempi . . . . .	53
21.3 Linee guida per comporre una tabella . . . . .	54
21.4 Compatibilità . . . . .	55
<b>22 ifpdf</b>	<b>56</b>
22.1 Esempi . . . . .	56
22.2 Pacchetti analoghi . . . . .	57
<b>23 enumitem</b>	<b>58</b>
23.1 Opzioni . . . . .	58
23.2 Comandi . . . . .	58
23.3 Chiavi e valori . . . . .	59
23.4 Esempi . . . . .	63

<b>24 url</b>	<b>65</b>
24.1 Opzioni . . . . .	65
24.2 Comandi . . . . .	65
24.3 Interazione con <code>hyperref</code> . . . . .	66
<b>25 tocloft</b>	<b>67</b>
25.1 Convenzioni . . . . .	67
25.2 Opzioni . . . . .	68
25.3 Comandi per modificare i titoli . . . . .	68
25.4 Comandi per la composizione degli indici . . . . .	69
25.5 Nuovi elenchi . . . . .	72
25.6 Compatibilità con <code>float</code> . . . . .	72
25.7 Compatibilità con altri pacchetti . . . . .	73
<b>26 graphicx</b>	<b>74</b>
26.1 Opzioni . . . . .	74
26.2 Documenti grafici esterni . . . . .	75
26.3 Comandi . . . . .	75
26.4 Esempi . . . . .	80
26.5 Avvertenze . . . . .	83
<b>27 color</b>	<b>84</b>
27.1 Opzioni . . . . .	84
27.2 Spazi di colore . . . . .	84
27.3 Definire colori . . . . .	85
27.4 Colorare il testo . . . . .	86
27.5 Colorare sfondi . . . . .	86
27.6 Tabella dei colori dello spazio <code>named</code> . . . . .	87
27.7 Limitazioni . . . . .	87
27.8 Altri pacchetti . . . . .	87
<b>28 fancyhdr</b>	<b>88</b>
28.1 Comandi . . . . .	88
28.2 Esempi . . . . .	90
<b>29 geometry</b>	<b>92</b>
29.1 Opzioni . . . . .	92
29.2 Esempi . . . . .	94
29.3 <code>\newgeometry</code> . . . . .	95
<b>30 emptypage</b>	<b>96</b>

## babel

```
\usepackage[⟨lingua⟩, …]{babel}
```

### 1.1 — Opzioni

L’ultima lingua specificata è quella principale del documento; le lingue vanno scelte nell’elenco che segue. Notare che `english` può significare `USenglish` oppure `UKenglish`, secondo quanto stabilito dal file `language.dat` della propria distribuzione. La lingua `english` è sempre disponibile, le altre vanno abilitate nella configurazione della distribuzione. Si noti che `ngerman` e `naustrian` usano la “Nuova Ortografia”.

acadian	croatian	indon	portuguese
afrikaans	czech	indonesian	romanian
albanian	danish	interlingua	russian
american	dutch	irish	samin
australian	english	italian	scottish
austrian	esperanto	latin	serbian
bahasa	estonian	lowersorbian	slovak
bahasai	finnish	magyar	slovene
bahasam	francais	malay	spanish
basque	french	meyalu	swedish
brazil	frenchb	naustrian	turkish
brazilian	galician	newzealand	ukrainian
breton	german	ngerman	upper-sorbian
british	germanb	norsk	welsh
bulgarian	greek	nynorsk	UKenglish
canadian	hebrew	polish	USenglish
canadien	hungarian	polutonikogreek	
catalan	icelandic	portuges	

Sebbene le opzioni possano essere dichiarate *globalmente*, si raccomanda, quando si dichiarino più lingue, di passarle esplicitamente a `babel`.

Il pacchetto `babel` può agire:

1. sulla sillabazione;
2. sulle convenzioni tipografiche;
3. sulle parole chiave e sulle date;
4. sulle abbreviazioni per comandi;
5. sui comandi specifici della lingua.

### 1.2 — Comandi

```
\selectlanguage{⟨lingua⟩}
```

La `⟨lingua⟩` deve essere stata specificata nelle opzioni date a `babel` (o fra le opzioni globali a `\documentclass`). Dal comando in poi è in vigore la lingua data come argomento, in tutti gli aspetti menzionati prima. È una dichiarazione *locale*, cioè vale solo nell’ambiente in cui si trova e non ai livelli superiori.

```
\foreignlanguage{⟨lingua⟩}{⟨testo⟩}
```

Compone il  $\langle testo \rangle$  nella  $\langle lingua \rangle$  dichiarata usando i comandi specifici, le abbreviazioni e la sillabazione.

```
\begin{otherlanguage}{⟨lingua⟩}
```

È equivalente a `\selectlanguage`, ma il cambiamento di lingua è locale all'ambiente stesso.

```
\begin{otherlanguage*}{⟨lingua⟩}
```

Comanda l'uso di sillabazione, convenzioni tipografiche, abbreviazioni e comandi specifici della  $\langle lingua \rangle$ , ma non agisce sulle parole chiave e sulle date.

```
\begin{hyphenrules}{⟨lingua⟩}
```

Abilita la sillabazione nella  $\langle lingua \rangle$ . Quasi tutte le distribuzioni definiscono una pseudo-lingua 'nohyphenation', nella quale non è definito alcuno schema di sillabazione. Dunque, dopo `\begin{hyphenrules}{nohyphenation}`, nessuna parola viene spezzata se non ci sono comandi di sillabazione espliciti.

```
\addto\captions⟨lingua⟩{⟨testo⟩}
```

Comanda di aggiungere alle definizioni specifiche della  $\langle lingua \rangle$  il  $\langle testo \rangle$ . Per esempio, volendo sostituire "Bibliografia" con "Letture suggerite", si può scrivere, nel preambolo:

```
\addto\captionsitalian{\%  
  \renewcommand{\bibname}{Letture suggerite}}
```

```
\language⟨nome⟩
```

Contiene il nome della lingua corrente.

```
\usesshorthands{⟨carattere⟩}
```

Abilita il  $\langle carattere \rangle$  come prefisso per le abbreviazioni.

```
\defineshorthand{⟨sequenza⟩}{⟨testo⟩}
```

Definisce  $\langle sequenza \rangle$  (uno o due caratteri) come abbreviazione per  $\langle testo \rangle$ .

```
\aliasshorthand{⟨carattere⟩}{⟨carattere⟩}
```

Definisce il primo  $\langle carattere \rangle$  come sinonimo del secondo. Il primo  $\langle carattere \rangle$  deve essere stato dichiarato nel preambolo usando `\usesshorthands`.

```
\languageshorthands{⟨lingua⟩}
```

Abilita le abbreviazioni definite nella  $\langle lingua \rangle$ , che deve essere stata dichiarata fra le opzioni date a babel.

```
\shorthandson | \shorthandsoff
```

Abilita (disabilita) le abbreviazioni. Per esempio, vanno usati prima e dopo di ambienti o comandi del pacchetto `xy` se la lingua corrente usa, per esempio, " come prefisso per le abbreviazioni.

```
\iflanguage{⟨lingua⟩}{⟨testo1⟩}{⟨testo2⟩}
```

Compone  $\langle testo_1 \rangle$  se la lingua corrente è  $\langle lingua \rangle$ , altrimenti  $\langle testo_2 \rangle$ . Si veda un esempio a pagina 17.



### 1.3 — Comandi specifici per l'italiano

Riportiamo i comandi per le parole chiave e il loro valore in italiano.

<code>\abstractname</code>	Sommario	<code>\indexname</code>	Indice analitico
<code>\alsoname</code>	vedi anche	<code>\listfigurename</code>	Elenco delle figure
<code>\appendixname</code>	Appendice	<code>\listtablename</code>	Elenco delle tabelle
<code>\bibname</code>	Bibliografia	<code>\pagename</code>	Pag.
<code>\ccname</code>	e p. c.	<code>\partname</code>	Parte
<code>\chaptername</code>	Capitolo	<code>\prefacename</code>	Prefazione
<code>\contentsname</code>	Indice	<code>\proofname</code>	Dimostrazione
<code>\enclname</code>	Allegati	<code>\refname</code>	Riferimenti bibliografici
<code>\figurename</code>	Figura	<code>\seename</code>	vedi
<code>\glossaryname</code>	Glossario	<code>\tablename</code>	Tabella
<code>\headtoname</code>	Per		

"

Inserisce un punto di sillabazione, utile per indicare sillabazioni etimologiche (per esempio, `iper"attivo`).

"|

Va usato nell'improbabile caso che la sillabazione etimologica sia richiesta subito prima di una lettera accentata.

""

Inserisce le virgolette alte aperte. Utile per chi non ha il carattere 'accento grave' sulla propria tastiera.

"< | ">

Inseriscono i 'caporali', cioè le virgolette francesi « e ». Se le si vuole usare, si raccomanda l'uso della codifica T1 con `\usepackage[T1]{fontenc}`.

"/

Equivalente a `\slash`, cioè una barra dopo la quale è ammesso spezzare la riga.

`\unit`

Utile per comporre unità di misura. Per esempio `$2\unit{m}$` produce '2 m'. Funziona sia in modo matematico che in modo testo.

`\ap{<testo>}` | `\ped{<testo>}`

Compongono il `<testo>` rispettivamente come apice o pedice. Per esempio, `7\ap{mo}` produce '7<sup>mo</sup>'. Meglio scrivere 'settimo'.

```
afterpage
\usepackage{afterpage}
```

```
\afterpage{<testo>}
```

Il `<testo>` viene eseguito appena la pagina corrente è stata composta e scritta nel documento DVI o PDF.

Per esempio, se si vuole lasciare una pagina completamente bianca dopo la pagina corrente, si può dare

```
\afterpage{\null\thispagestyle{empty}\clearpage}
```

**array**  
`\usepackage{array}`

Ogni estensione qui indicata vale sia per l'ambiente `tabular` che per `array`; questi ambienti hanno un argomento obbligatorio che consiste di una successione di specificatori per ciascuna colonna. È possibile ripetere una successione di specificatori  $\langle spec \rangle$  un certo numero  $\langle num \rangle$  di volte con la costruzione  $*\langle num \rangle\{\langle spec \rangle\}$ ; per esempio,  $*\{3\}\{c\}$  equivale a una specificazione `ccc`.

### 3.1 — Specificatori

`l`

Specifica una colonna con allineamento a sinistra (come in  $\text{\LaTeX}$  normale).

`c`

Specifica una colonna con allineamento al centro (come in  $\text{\LaTeX}$  normale).

`r`

Specifica una colonna con allineamento a destra (come in  $\text{\LaTeX}$  normale).

`p{\langle dimen \rangle}`

Specifica una colonna in cui le celle sono composte come paragrafi di larghezza  $\langle dimen \rangle$  ( $\text{\LaTeX}$  normale). La cella è allineata alla riga a cui appartiene rispetto alla linea di base superiore (come in  $\text{\LaTeX}$  normale). È analogo a scrivere `\parbox[t]{\langle dimen \rangle}`.

`m{\langle dimen \rangle}`

Specifica una colonna in cui le celle sono composte come paragrafi di larghezza  $\langle dimen \rangle$ . La cella è centrata rispetto alla riga a cui appartiene (`array`). È analogo a scrivere `\parbox[c]{\langle dimen \rangle}`.

`b{\langle dimen \rangle}`

Specifica una colonna in cui le celle sono composte come paragrafi di larghezza  $\langle dimen \rangle$ . La cella è allineata alla riga a cui appartiene rispetto alla linea di base inferiore (`array`). È analogo a scrivere `\parbox[b]{\langle dimen \rangle}`.

`@{\langle testo \rangle}`

Specifica che fra le colonne in cui si trova sia composto il  $\langle testo \rangle$ , senza interporre l'usuale spazio fra le colonne (come in  $\text{\LaTeX}$  normale).

`!\{\langle testo \rangle\}`

Specifica che fra le colonne in cui si trova sia composto il  $\langle testo \rangle$ , interponendo anche l'usuale spazio fra le colonne (`array`).

`|`

Specifica che fra le colonne in cui si trova sia composta una linea verticale (come in  $\text{\LaTeX}$  normale).

`>{\testo}`

Può essere usato prima di uno specificatore `l`, `c`, `r`, `p`, `m` o `b`. Indica che prima del contenuto di ogni cella della colonna sia eseguito il `<testo>` (si vedano gli esempi con `\newcolumnntype`).

`<{\testo}`

Può essere usato dopo uno specificatore `l`, `c`, `r`, `p`, `m` o `b`. Indica che dopo il contenuto di ogni cella della colonna sia eseguito il `<testo>` (si vedano gli esempi con `\newcolumnntype`).

### 3.2 — Comandi

`\extrarowheight`

È un parametro dimensionale che può essere utile per migliorare le tabelle. All'altezza di ogni cella viene aggiunta questa dimensione. Per esempio,

```
\setlength{\extrarowheight}{1pt}
```

È consigliabile che l'aggiustamento sia fatto tabella per tabella.

`\newcolumnntype{<carattere>}[<num>]{<testo>}`

Il `<carattere>` deve essere uno non già usato come specificatore di colonne nei preamboli delle tabelle. Il `<testo>` non può essere arbitrario, ma va composto nel modo seguente:

```
[>{\prima}] <spec> [<{\dopo}]
```

(le parentesi quadre indicano che quella parte può essere omessa). Con `<spec>` si intende uno specificatore di colonna, eventualmente definito anch'esso con `\newcolumnntype`. Per esempio,

```
\newcolumnntype{C}{>{\$}c<{\$}}
```

definisce uno specificatore di colonna che produce una colonna centrata in modo matematico in `tabular` e in modo testo in `array`. Infatti `<prima>` viene inserito prima del contenuto della cella e, analogamente, `<dopo>` viene inserito alla fine. Dato quel comando nel preambolo, il codice

```
\begin{tabular}{l*{5}{|C}}
& 0 & \pi/2 & \pi & 3\pi/2 & 2\pi \\
\cline{2-6}
Valore del seno & 0 & 1 & 0 & -1 & 0 \\
Valore del coseno & 1 & 0 & -1 & 0 & 1 \\
\end{tabular}
```

produrrà la tabella

	0	$\pi/2$	$\pi$	$3\pi/2$	$2\pi$
Valore del seno	0	1	0	-1	0
Valore del coseno	1	0	-1	0	1

senza bisogno di specificare il modo matematico in ciascuna cella. In `<prima>` e `<dopo>` può andare qualunque cosa.

L'argomento opzionale `<num>` è il numero di argomenti; si può usare la stessa sintassi di `\newcommand`, indicando i parametri da sostituire con `#1`, `#2` e così via. Diamo di seguito un esempio; ovviamente si possono trovare applicazioni migliori.

```
\newcolumnntype{v}[1]{>{\#1: \hfill}1}
\begin{tabular}{v{Nazione}|v{Valore}}
Italia & 640 \\
Germania & 231 \\
Francia & 100 \\
Turchia & 91 \\
Spagna & 1003 \\
\end{tabular}
```

Nazione:	Italia	Valore:	640
Nazione:	Germania	Valore:	231
Nazione:	Francia	Valore:	100
Nazione:	Turchia	Valore:	91
Nazione:	Spagna	Valore:	1003

`\showcols`

Scrive a terminale e sul LOG la lista degli specificatori definiti con `\newcolumnntype`.

`\firstline`

Va usato al posto di `\hline` per una linea orizzontale sopra la tabella, permettendo di allineare la tabella rispetto alla riga di base superiore (usando `\begin{tabular}[t]`).

`\lastline`

Va usato al posto di `\hline` per una linea orizzontale sotto la tabella, permettendo di allineare la tabella rispetto alla riga di base inferiore (usando `\begin{tabular}[b]`).

`\tabularnewline`

Equivalente al comando `\\` di fine riga dell'allineamento.

`\arraybackslash`

Supponiamo di volere una tabella in cui l'ultima colonna sia formata da paragrafi sbandierati a destra. La soluzione di scrivere

```
\begin{tabular}{...>\raggedright}p{4cm}}
```

non funziona, perché il comando `\\` viene ridefinito da `\raggedright`. Potremmo terminare ogni cella con il comando `\tabularnewline`, ma c'è un modo migliore:

```
\begin{tabular}{...>\raggedright\arraybackslash}p{4cm}}
```

cura il problema, perché `\arraybackslash` ridà il valore corretto a `\\` dopo la ridefinizione di `\raggedright`. Naturalmente, gli eventuali capo riga manuali nella cella devono essere dati con `\newline`.

**bm**  
`\usepackage{bm}`

Non si abusi della possibilità di scrivere simboli matematici in carattere nero fornita da questo pacchetto. La tradizione tipografica dice che i caratteri neri usati come simboli matematici sono diritti (con `\mathbf`); le normative UNI-ISO dicono altrimenti.

#### 4.1 — Comandi

`\bm{⟨formula⟩}`

Scrive la  $\langle formula \rangle$  (che deve essere materiale adatto al modo matematico) in carattere nero. Per esempio,

`\[ \alpha \neq \bm{\alpha} \]`

produce

$\alpha \neq \alpha$

Il comando può risultare utile quando si debba comporre una formula in un titolo di capitolo o sezione e la classe usi il nero per queste parti.

`\DeclareBoldMathCommand[⟨versione⟩]{⟨comando⟩}{⟨formula⟩}`

Il  $\langle comando \rangle$  non deve essere già definito. Dopo questo,  $\langle comando \rangle$  comporrà la  $\langle formula \rangle$  (tipicamente un solo simbolo) come se fosse `\bm{⟨formula⟩}`. Per esempio

`\DeclareBoldMathCommand{\balpha}{\alpha}`

risulta più efficiente di scrivere ogni volta `\bm{\alpha}`. L'argomento  $\langle versione \rangle$  opzionale, se omissso, vale `bold`; se è disponibile una versione `heavy` (poche famiglie di caratteri ce l'hanno) si può usare quella. È sconsigliabile usare la versione `normal`. (Esercizio: perché?)

`\bmdefine{⟨comando⟩}{⟨formula⟩}`

È equivalente a `\DeclareBoldMathCommand[bold]{⟨comando⟩}{⟨formula⟩}`.

`\hmdefine{⟨comando⟩}{⟨formula⟩}`

È equivalente a `\DeclareBoldMathCommand[heavy]{⟨comando⟩}{⟨formula⟩}`.

#### 4.2 — Esempi

È bene fare attenzione. Scrivendo

```
$1 g \bm{g}$
$2 \mathrm{g} \bm{g}$
$3 {g} \bm{{g}}$
$4 \mathrm{{g}} \bm{{g}}$
$5 \mathrm{g} \bm{\mathrm{g}}$
```

si ottiene

**1gg 2gg 3gg 4gg 5gg**

## calc

```
\usepackage{calc}
```

### 5.1 — Introduzione

I calcoli aritmetici con  $\text{\LaTeX}$  non sono molto facili e richiedono spesso il passaggio attraverso registri temporanei. Il pacchetto `calc` fornisce strumenti per facilitare i calcoli diretti.

Supponiamo, per esempio, di avere una figura di larghezza non troppo grande che vogliamo inserire in modo che occupi tutta una riga. Tuttavia non desideriamo che diventi troppo grande e l'altezza superi quella della gabbia tipografica. Siccome la figura è generata da un programma esterno, non ne conosciamo a priori le dimensioni.

Il problema diventa complicato perché in  $\text{\TeX}$  non è ammesso dividere per una dimensione (si può fare, ma con un trucco da maghi). Ecco come si può operare con `calc`:

```
\newdimen{\Xsize}\newdimen{\Ysize}
\newdimen{\finalYsize}
\setlength{\Xsize}{\widthof{\includegraphics{figura}}}
\setlength{\Ysize}{\heightof{\includegraphics{figura}}}
\setlength{\finalYsize}{\Ysize*\ratio{\textwidth}{\Xsize}}
```

A questo punto è possibile confrontare `\finalYsize` con una dimensione prefissata (per esempio `\textheight`) e operare di conseguenza (si veda come nella descrizione di `ifthen`).

### 5.2 — Operazioni

Le operazioni ammesse sono somma, sottrazione, moltiplicazione e divisione con interi o reali a virgola mobile. In certi casi il risultato è soggetto al troncamento della parte decimale.

Scriviamo la sintassi dei comandi ammessi.

```
<espressione tipo> → <termine tipo>
| <espressione tipo><più o meno><termine tipo>
<termine tipo> → <fattore tipo>
| <termine tipo><per o diviso><fattore intero>
| <termine tipo><per o diviso><numero reale>
<fattore tipo> → <tipo>
| <fattore dimensionale> | (<espressione tipo>)
<fattore dimensionale> → <comando dimensionale>{<testo>}
<comando dimensionale> → \widthof | \heightof | \depthof
<più o meno> → + | -
<per o diviso> → * | /
<numero reale> → \real{<costante decimale>}
| \ratio{<espressione dimensionale>}{<espressione dimensionale>}
```

In questa descrizione formale, *tipo* sta per 'intero' (o 'intera') oppure 'dimensionale'. Per esempio, le prime due righe descrivono il significato sia di `<espressione intera>` sia di `<espressione dimensionale>`. Per essere più espliciti, queste sono espressioni intere:

```
2 * (3 + 5) -4
6 + 3
1*4+2*3
```

mentre queste sono espressioni dimensionali:

```
2cm - 3pt + 4cm*\real{1.3} - \textwidth + \widthof{pippo}
\textheight*\ratio{\textwidth}{6dd}
```

Dove è richiesta una dimensione se ne può dare una esplicita oppure una implicita con uno dei parametri dimensionali di L<sup>A</sup>T<sub>E</sub>X. Si noti che `\real` e `\ratio` devono comparire sempre *dopo* l'operatore di moltiplicazione o divisione, e si può solo moltiplicare una dimensione per un numero, non viceversa. Gli spazi in un'espressione sono ignorati e possono utilmente essere impiegati per aumentare la leggibilità.

Ogni volta che L<sup>A</sup>T<sub>E</sub>X deve leggere una dimensione o un numero intero, si può usare una di queste espressioni, cioè con `\setcounter` o `\setlength`. La cosa funziona anche con i comandi o gli ambienti che richiedono una dimensione, come `\makebox`, `\parbox` o `minipage`.

### 5.3 — Comandi

```
\real{costante decimale}
```

Un qualunque numero decimale esplicito; il separatore decimale deve essere il punto.

```
\ratio{espressione dimensionale}{espressione dimensionale}
```

È spiegato nella descrizione formale delle operazioni.

```
\widthof | \heightof | \depthof{testo}
```

Ciascuno dei comandi prende ad argomento un *testo* che viene composto (e poi scartato). Il primo calcola la larghezza del risultato, il secondo l'altezza, il terzo la profondità, cioè di quanto il testo si estende sotto la linea di base.

```
\widthof{p} → 5.5542 pt
\heightof{p} → 4.3045 pt
\depthof{p} → 1.94397 pt
```

Per fare un esempio meno banale: si vuole comporre una `minipage` larga come una riga di un testo variabile, aumentato di un centimetro.

```
\newenvironment{varmp}[1]
  {\begin{minipage}{\widthof{#1}+1cm}}
  {\end{minipage}}
```



**dcolumn**  
`\usepackage{dcolumn}`

Lo scopo di `dcolumn` è di facilitare l'inserimento di cifre con parte decimale in una tabella (`tabular` o `array`). Il pacchetto carica automaticamente `array`.

`D{<sep-in>}{<sep-out>}{<numero>}`

È uno specificatore di colonne con tre argomenti. Il primo è il carattere da usare nel documento `.tex` per indicare la separazione fra le colonne, il secondo quello usato nel documento composto. Il terzo argomento è il numero di cifre decimali. I numeri saranno allineati rispetto al separatore della parte frazionaria; nel caso il terzo argomento sia negativo, il separatore sarà al centro della colonna. Si veda l'osservazione alla fine del capitolo.

### 6.1 — Esempi

Supponiamo di voler comporre una tabella a tre colonne, nella quale la seconda e la terza consistono di numeri decimali; il numero massimo di cifre decimali nella seconda è 3, nella terza non è specificato.

```
\begin{tabular}
  {lD{.}{,}{3}D{.}{\cdot}{-1}|}
Italia & 640.3 & 1.22 \\
Germania & 231.12 & 2 \\
Francia & 100.1 & 3.12334 \\
Turchia & 91.399 & 90 \\
Spagna & 1003.121 & 10.03
\end{tabular}
```

Italia	640,3	1.22
Germania	231,12	2
Francia	100,1	3.12334
Turchia	91,399	90
Spagna	1003,121	10.03

Si noti come viene trattata correttamente la mancanza di una parte frazionaria.

Facciamo un altro esempio, per mostrare come si possa scrivere una tabella che funzioni con lo stesso codice in italiano e in inglese, dove si usano separatori decimali diversi. Nel codice che segue si vede che come `<sep-in>` si può usare un carattere qualunque.

```
\newcolumntype{U}{D{?}{\iflanguage{english}{.}{,}}{2}}
\newcommand{\tabella}{\begin{tabular}{|l|U|}
  Germania & 2?31 \\
  Spagna & 100?3
\end{tabular}}
\selectlanguage{english}\tabella
\quad
\selectlanguage{italian}\tabella
```

Germania	2.31	Germania	2,31
Spagna	100.3	Spagna	100,3

### 6.2 — Osservazione

Nel terzo argomento è possibile scrivere `<numero><carattere><numero>` per indicare il massimo numero di cifre nella parte intera e nella parte frazionaria. Questo è utile nel caso si desideri intestare la colonna impiegando un comando `\multicolumn`. Qui `<carattere>` indica un qualunque carattere che non sia una cifra; si consiglia di usare un punto.

**enumerate**  
`\usepackage{enumerate}`

### 7.1 — Descrizione

Talvolta è utile avere a disposizione la possibilità di costruirsi liste numerate particolari. Il meccanismo normale messo a disposizione da  $\LaTeX$  permette solo di cambiare globalmente il modo di comporre le liste numerate, con ottime ragioni. D'altra parte, non tutte le liste sono uguali; si potrebbe usare l'argomento opzionale di `\item`, ma questa sarebbe una forzatura che costringerebbe a scrivere esplicitamente i numeri. Si veda però il capitolo 23 per un pacchetto più potente.

Il pacchetto `enumerate` aggiunge all'ambiente `enumerate` un argomento opzionale che determina il modo di comporre l'etichetta numerata delle liste in modo automatico. Il pacchetto riserva cinque caratteri speciali: '1' 'i' 'I' 'a' 'A'.

`1 | i | I | a | A`

Indica che il numero sia composto, rispettivamente, con cifre arabiche, numeri romani minuscoli, numeri romani maiuscoli, lettere minuscole o lettere maiuscole.

`\label`

Menzioniamo qui il comando anche se non è modificato. Il comando `\label` può essere dato dopo un `\item`, ma genera solo *il numero* (nella forma indicata, cioè in cifre arabiche, come numero romano o come lettera). Il prefisso  $\langle testo_1 \rangle$  e il suffisso  $\langle testo_2 \rangle$  vanno aggiunti esplicitamente con

$\langle testo_1 \rangle \backslash \text{ref} \{ \langle etichetta \rangle \} \langle testo_2 \rangle$ .

### 7.2 — Uso

Il comando di inizio dell'ambiente `enumerate` può essere seguito da un argomento opzionale

`\begin{enumerate} [ \langle testo_1 \rangle \langle carattere \rangle \langle testo_2 \rangle ]`

dove  $\langle carattere \rangle$  è uno dei cinque detti in precedenza, che non deve comparire 'nascosto' tra parentesi graffe. L'etichetta che precede ciascun elemento della lista sarà composta come

$\langle testo_1 \rangle \langle numero \rangle \langle testo_2 \rangle$

e il  $\langle numero \rangle$  è come stabilito dalla corrispondenza descritta sopra. Se il carattere indicato è 'a' oppure 'A', la lista non può avere più di 26 elementi. Eventuali dichiarazioni contenute nel prefisso o nel suffisso sono locali all'etichetta.

Primo esempio: vogliamo comporre una lista che enunci le proprietà richieste a una struttura per farne uno spazio vettoriale; per comodità del lettore vogliamo che i numeri siano preceduti dalla sigla 'SV' e il tutto sia fra parentesi tonde.

`\begin{enumerate} [(SV1)]`

Secondo esempio: vogliamo scrivere parecchie liste del tipo visto nel primo esempio, con varie sigle. Definiremo un ambiente appropriato cui daremo come argomento la sigla.

```
\newenvironment{assiomi}[1]
  {\begin{enumerate}[({#1}1)]}
  {\end{enumerate}}
```

Si noti come il riferimento al primo argomento, cioè #1, viene interpretato correttamente, ma viene messo tra parentesi graffe per evitare che eventuali caratteri dei cinque speciali possano dare problemi. La cosa non era necessaria nel primo esempio, perché le lettere ‘S’ e ‘V’ non sono fra i caratteri riservati.

Terzo esempio: vogliamo definirci liste personalizzate per indicare (1) condizioni che seguono da un’ipotesi; (2) condizioni equivalenti; (3) condizioni da soddisfare per una definizione. Vogliamo indicare le prime con cifre arabe, le seconde con lettere minuscole, le terze con lettere maiuscole, tra parentesi tonde. Cerchiamo di farlo nel modo più astratto possibile: trattandosi di un articolo da inviare per la pubblicazione, dobbiamo essere in grado di modificare la resa tipografica senza agire sul documento, ma solo sul preambolo. Una piccola complicazione è che le etichette devono apparire con carattere diritto anche in ambiente corsivo.

```
\newenvironment{roster}[1]
  {\begin{enumerate}[\upshape(#1)]}
  {\end{enumerate}}
\newenvironment{concl}{\begin{roster}{1}}{\end{roster}}
\newenvironment{eqcond}{\begin{roster}{a}}{\end{roster}}
\newenvironment{defcond}{\begin{roster}{A}}{\end{roster}}
```

Si noti che è stato definito un nuovo ambiente generico e poi i tre ambienti particolari sono stati definiti tramite quello. Se la rivista ci dovesse chiedere una modifica, per esempio di racchiudere i numeri o le lettere tra parentesi quadre, sarà sufficiente agire su una sola riga del documento. L’effetto della dichiarazione `\upshape` non si propaga oltre la composizione dell’etichetta.

Quarto esempio: vogliamo che il numero dell’elemento compaia a pedice della sigla, ma abbiamo il problema che il numero degli elementi può essere superiore a nove. Scrivendo

```
\begin{enumerate}[(SV$_{1}$)]
```

il carattere ‘1’ *non* viene interpretato come speciale. Scrivendo

```
\begin{enumerate}[(SV$_1$)]
```

il decimo elemento verrebbe etichettato come ‘(SV<sub>10</sub>)’ e così sarebbe per i successivi. La soluzione è di scrivere

```
\begin{enumerate}[(SV$_\bgroup1\egroup$)]
```

Il trucco funziona perché `enumerate` non riconosce i comandi `\bgroup` e `\egroup` come parentesi graffe, ma per L<sup>A</sup>T<sub>E</sub>X la costruzione `$a_\bgroup10\egroup$` è equivalente a `$a_{10}$`.

**ifthen**  
`\usepackage{ifthen}`

In molte situazioni è conveniente poter impiegare comandi che hanno significati diversi in contesti diversi. Il pacchetto `ifthen` fornisce un'interfaccia ai comandi primitivi di  $\TeX$  che riguardano i condizionali e le variabili booleane.

### 8.1 — Comandi

`\ifthenelse{<test>}{<vero>}{<falso>}`

Valuta  $\langle test \rangle$ , che restituisce ‘vero’ o ‘falso’ ed esegue di conseguenza  $\langle vero \rangle$  o  $\langle falso \rangle$ .

`\whiledo{<test>}{<comandi>}`

Esegue  $\langle comandi \rangle$  fino a che il  $\langle test \rangle$  restituisce il valore ‘vero’.

`\newboolean{<nome>}`

Crea una variabile booleana alla quale ci si riferisce con  $\langle nome \rangle$  che deve essere una successione di lettere. Se la variabile esiste già, il comando produce un errore.

`\provideboolean{<nome>}`

Crea una variabile booleana alla quale ci si riferisce con  $\langle nome \rangle$  che deve essere una successione di lettere. È ignorato se la variabile esiste già.

`\setboolean{<nome>}{<valore>}`

Assegna alla variabile booleana  $\langle nome \rangle$  il valore  $\langle valore \rangle$  che può essere solo `true` oppure `false`. Quando una variabile è creata, le viene assegnato il valore `false`.

### 8.2 — I test

I  $\langle test \rangle$  sono costruiti a partire dalle espressioni elementari che elenchiamo di seguito.

`<numero><op><numero>`

Espressione elementare per costruire un  $\langle test \rangle$ . Ciascun  $\langle numero \rangle$  può essere un intero esplicito o implicito (tramite `\value{<contatore>}` o `\pageref{<etichetta>}`);  $\langle op \rangle$  deve essere `<`, `=` o `>`. Restituisce ‘vero’ se i due interi sono nella relazione indicata. Dà errore se non si usano numeri.

`\isodd{<numero>}`

Espressione elementare per costruire un  $\langle test \rangle$ . Restituisce ‘vero’ se il  $\langle numero \rangle$  è dispari. Il  $\langle numero \rangle$  può essere un intero esplicito oppure `\value{<contatore>}` o anche `\pageref{<etichetta>}`. Restituisce ‘falso’ altrimenti (anche se  $\langle numero \rangle$  non è un numero, per esempio quando un riferimento a una pagina non è ancora noto). Attenzione: `\isodd{11xx}` restituisce vero!

`\isundefined{<comando>}`

Espressione elementare per costruire un  $\langle test \rangle$ . Restituisce ‘vero’ se il  $\langle comando \rangle$  non è definito.

```
\equal{<stringa>}{<stringa>}
```

Espressione elementare per costruire un  $\langle test \rangle$ . Restituisce ‘vero’ se le due stringhe sono uguali.

```
\lengthtest{<dimen><op><dimen>}
```

Espressione elementare per costruire un  $\langle test \rangle$ . Il primo e il terzo argomento devono essere lunghezze (o anche parametri che contengono lunghezze, come `\parindent`);  $\langle op \rangle$  deve essere  $<, = o >$ . Restituisce ‘vero’ se le lunghezze soddisfano la relazione indicata.

```
\boolean{<nome>}
```

Espressione elementare per costruire un  $\langle test \rangle$ . Restituisce il valore della variabile booleana  $\langle nome \rangle$ .

Un  $\langle test \rangle$  si ottiene combinando espressioni elementari tramite i comandi `\AND`, `\OR` e `\NOT` e le ‘parentesi’ `\(` e `\)`. Si possono usare anche `\and`, `\or` e `\not`, ma è consigliabile usare le prime forme per maggiore leggibilità.

```
\( \isodd{ \value{page} } \AND \value{page} > 31 \) \OR \value{page} < 10
```

Questo  $\langle test \rangle$  viene valutato ‘vero’ se siamo a una pagina dispari dopo la pagina 31 oppure in una pagina prima della decima.

L’esempio non è dei migliori, ma è utile per illustrare un piccolo problema: il numero di pagina corrente può non rispecchiare quello ‘reale’ perché  $\text{\LaTeX}$  compone sempre più materiale prima di stabilire come formare le pagine. Per riferirsi a un numero di pagina, è meglio usare il sistema di assegnare un’etichetta con `\label`.

Un’altra avvertenza: la parte  $\langle comandi \rangle$  usata in `\whiledo` deve modificare le variabili che vengono confrontate nel  $\langle test \rangle$ . Il comando `\whiledo{1<2}{a}` produce un ciclo infinito. Per comporre una semplice tabella che contiene i caratteri del *font* corrente si può usare

```
\setlength{\dimen0}{.125\linewidth}
\newcounter{cc}
\noindent
\whiledo{\value{cc}<256}
  {%
    \makebox[\dimen0][l]{%
      \arabic{cc}$=\symbol{\value{cc}}%
      \hspace{0pt}\stepcounter{cc}%
    }%
  }
```

A ogni ciclo viene composta una scatola di ampiezza un ottavo della larghezza della riga (`\dimen0` è un parametro di lunghezza impiegabile per lavoretti di poco conto), con allineamento a sinistra; tra una scatola e l’altra non c’è spazio, ma  $\text{\LaTeX}$  può spezzare le righe (`\hspace{0pt}`). A ogni ciclo il valore del contatore viene aumentato di uno; quando il contatore assume il valore 256, il ciclo si interrompe. Si veda un esempio (modificato) a pagina 22 (non c’entra niente con il capitolo in cui si trova, è solo per riempire uno spazio altrimenti vuoto).

**indentfirst**  
`\usepackage{indentfirst}`

Il pacchetto non definisce alcun nuovo comando; il suo unico effetto è di imporre il rientro (*indent*) anche al primo paragrafo dopo un comando di sezione come `\chapter` o `\section`, secondo l'uso europeo continentale.

La tabella qui sotto è relativa al capitolo precedente, ma serve anche a riempire lo spazio su questa pagina.

**Tabella 9.1** Esempio di tabella dei caratteri, usando Zapf Dingbats

0 =	1 =	2 =	3 =	4 =	5 =	6 =	7 =
8 =	9 =	10 =	11 =	12 =	13 =	14 =	15 =
16 =	17 =	18 =	19 =	20 =	21 =	22 =	23 =
24 =	25 =	26 =	27 =	28 =	29 =	30 =	31 =
32 =	33 = ☞	34 = ☜	35 = ☚	36 = ☛	37 = ☞	38 = ☺	39 = ☻
40 = ☛	41 = ☞	42 = ☜	43 = ☚	44 = ☛	45 = ☞	46 = ☞	47 = ☞
48 = ☞	49 = ☞	50 = ☞	51 = ✓	52 = ✓	53 = ✕	54 = ✕	55 = ✕
56 = ✕	57 = ☞	58 = ☞	59 = ☞	60 = ☞	61 = †	62 = †	63 = †
64 = ☞	65 = ☞	66 = ☞	67 = ☞	68 = ☞	69 = ☞	70 = ◆	71 = ◇
72 = ★	73 = ☆	74 = ☼	75 = ☆	76 = ★	77 = ★	78 = ★	79 = ★
80 = ☆	81 = ✱	82 = ✱	83 = ✱	84 = ✱	85 = ✱	86 = ✱	87 = ✱
88 = ✱	89 = ✱	90 = ✱	91 = ✱	92 = ✱	93 = ✱	94 = ✱	95 = ✱
96 = ✱	97 = ✱	98 = ✱	99 = ✱	100 = ✱	101 = ✱	102 = ✱	103 = ✱
104 = ✱	105 = ✱	106 = ✱	107 = ✱	108 = ●	109 = ○	110 = ■	111 = □
112 = □	113 = □	114 = □	115 = ▲	116 = ▼	117 = ◆	118 = ❖	119 = ►
120 =	121 =	122 =	123 = ‘	124 = ’	125 = “	126 = ”	127 =
128 =	129 =	130 =	131 =	132 =	133 =	134 =	135 =
136 =	137 =	138 =	139 =	140 =	141 =	142 =	143 =
144 =	145 =	146 =	147 =	148 =	149 =	150 =	151 =
152 =	153 =	154 =	155 =	156 =	157 =	158 =	159 =
160 =	161 = ♡	162 = ♡	163 = ♡	164 = ♥	165 = ♡	166 = ♡	167 = ♡
168 = ♣	169 = ♦	170 = ♥	171 = ♠	172 = ①	173 = ②	174 = ③	175 = ④
176 = ⑤	177 = ⑥	178 = ⑦	179 = ⑧	180 = ⑨	181 = ⑩	182 = ①	183 = ②
184 = ③	185 = ④	186 = ⑤	187 = ⑥	188 = ⑦	189 = ⑧	190 = ⑨	191 = ⑩
192 = ①	193 = ②	194 = ③	195 = ④	196 = ⑤	197 = ⑥	198 = ⑦	199 = ⑧
200 = ⑨	201 = ⑩	202 = ①	203 = ②	204 = ③	205 = ④	206 = ⑤	207 = ⑥
208 = ⑦	209 = ⑧	210 = ⑨	211 = ⑩	212 = →	213 = →	214 = ↔	215 = ↑
216 = ↘	217 = →	218 = ↗	219 = ➔	220 = ➔	221 = →	222 = →	223 = ➔
224 = ➔	225 = ➔	226 = ➔	227 = ➔	228 = ➔	229 = ➔	230 = ➔	231 = ➔
232 = ➔	233 = ➔	234 = ➔	235 = ➔	236 = ➔	237 = ➔	238 = ➔	239 = ➔
240 =	241 = ➔	242 = ➔	243 = ➔	244 = ➔	245 = ➔	246 = ➔	247 = ➔
248 = ➔	249 = ➔	250 = ➔	251 = ➔	252 = ➔	253 = ➔	254 = ➔	255 =

**layout**

```
\usepackage[⟨opzione⟩, …]{layout}
```

**10.1 — Opzioni**

Un primo gruppo di opzioni permette di scegliere la lingua per le descrizioni.

brazilian	francais	italian	spanish
dutch	french	ngerman	
english	german	portuguese	

```
silent | verbose
```

La prima è quella normale; con la seconda i dati appaiono sul terminale e nel LOG.

```
integers | reals
```

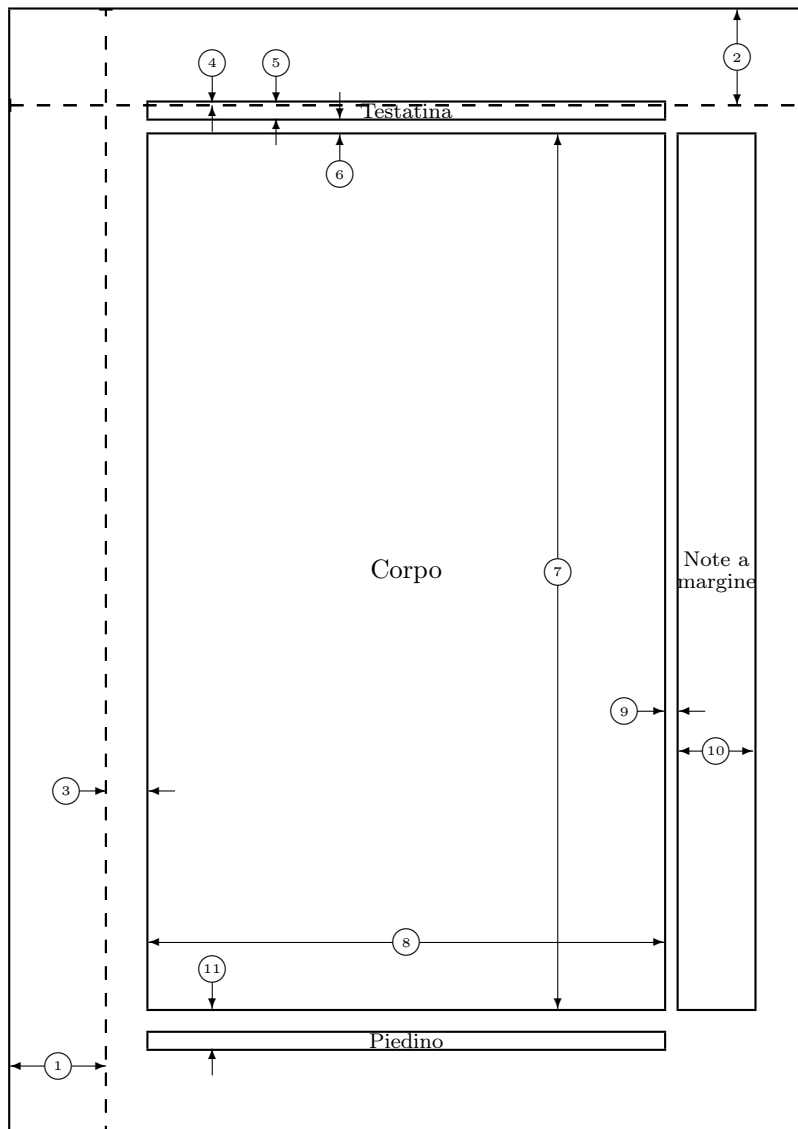
La prima è quella normale; con la seconda i valori delle dimensioni vengono espressi con la parte decimale invece che essere troncati.

```
\layout
```

L'unico comando definito da questo pacchetto produce uno schema della forma della pagina, con indicate le dimensioni dei vari elementi: gabbia, margini, altezza e distanza della testatina e così via. Se il documento è scritto in una classe che prevede l'opzione `twoside`, sono prodotti due schemi per le pagine pari e dispari. Tutte le dimensioni sono mostrate in punti tipografici.

```
\layout*
```

Il comando `\layout` tiene in memoria i vari parametri. Se si sono modificati i valori, si può usare `\layout*` per ricalcolare lo schema.



1	<code>un pollice + \hoffset</code>	2	<code>un pollice + \voffset</code>
3	<code>\oddsidemargin = 32.25894pt</code>	4	<code>\topmargin = -2.74657pt</code>
5	<code>\headheight = 12.0pt</code>	6	<code>\headsep = 12.0pt</code>
7	<code>\textheight = 658.0pt</code>	8	<code>\textwidth = 373.45pt</code>
9	<code>\marginparsep = 11.0pt</code>	10	<code>\marginparwidth = 57.0pt</code>
11	<code>\footskip = 30.0pt</code>		<code>\marginparpush = 5.0pt (non mostrato)</code>
	<code>\hoffset = 0.0pt</code>		<code>\voffset = 0.0pt</code>
	<code>\paperwidth = 597.50787pt</code>		<code>\paperheight = 845.04684pt</code>



## longtable

`\usepackage[opzione]{longtable}`

### 11.1 — Opzioni

#### errorshow

Al terminale vengono mostrati solo gli errori nella composizione di una tabella e gli avvertimenti compaiono solo nel LOG.

#### pausing

Ogni avvertimento si comporta come un errore; è utile per il ‘debugging’ di una tabella particolarmente complicata.

### 11.2 — Descrizione

In molte situazioni ci si trova a dover comporre una tabella troppo grande per una sola pagina: si pensi a un lungo elenco di variabili con i valori preassegnati o alla lista delle nazioni indipendenti con vari dati per ciascuna.

Il pacchetto `longtable` può risolvere il problema. Ne esiste uno simile, chiamato `supertabular`, ma non è del tutto compatibile con `array` e altri pacchetti.

Una tabella che occupa più pagine deve essere composta in modo asincrono. Il pacchetto raccoglie a ogni compilazione le larghezze delle celle di ogni pagina e alla successiva cerca di usare le larghezze giuste su tutte le pagine. Possono essere necessarie alcune compilazioni successive per ottenere il risultato finale.

Ogni tabella composta con `longtable` si compone di varie parti:

- un’intestazione iniziale;
- un’intestazione normale;
- un piede finale;
- un piede normale;
- il corpo della tabella;
- le didascalie.

L’intestazione iniziale può coincidere con l’intestazione normale, così come il piede finale può coincidere con il piede normale. Ciascuno di questi elementi può mancare, ma è consigliabile che compaiano almeno le intestazioni o i piedi per aiutare il lettore a capire che la stessa tabella occupa più di una pagina.

Le didascalie, se presenti, possono essere uguali in ciascuna pagina o diverse sulla prima e quelle successive. Vanno specificate insieme all’intestazione (e possono sostituirla).

La struttura di una `longtable` è descritta nella tabella 11.1 a pagina 28; si veda più avanti per la sintassi precisa del comando `\caption`. Il pacchetto compone la tabella un certo numero di righe alla volta, specificate nel contatore `LTchunks`. Se si verificano problemi di memoria, si può provare a diminuirne il valore (con `\setcounter`); occorre però tenere presente che tale valore *deve* essere maggiore del numero di righe nell’intestazione e nel piede.

Ricordiamo che non è obbligatorio che sia presente la didascalia. In questo caso è bene specificare un’intestazione; quella iniziale e quella normale possono coincidere, ma di solito si adotta una strategia del tipo

```
\hline
<intestazione iniziale>\
\hline\endfirsthead
\hline
<intestazione iniziale> (segue)\
\hline\endhead
```

### 11.3 — L'ambiente longtable

```
\begin{longtable}[<carattere>]{<preambolo>}
```

Si veda più avanti per l'argomento opzionale. Il `<preambolo>` è costruito esattamente come per `tabular` e si possono usare le estensioni messe a disposizione da `array` se questo è stato caricato.

### 11.4 — Parametri

```
\LTleft
```

Spazio a sinistra della tabella. Lunghezza elastica, valore usuale `\fill`.

```
\LTright
```

Spazio a destra della tabella. Lunghezza elastica, valore usuale `\fill`.

```
\LTpre
```

Spazio verticale prima della tabella. Lunghezza elastica, valore usuale `\bigskipamount`.

```
\LTpost
```

Spazio verticale dopo la tabella. Lunghezza elastica, valore usuale `\bigskipamount`.

```
\LTcapwidth
```

Larghezza della `\parbox` che contiene la didascalia. Valore usuale 4 in.

```
LTchunksize
```

Numero di righe per blocco. Contatore, valore usuale 20.

### 11.5 — Argomenti opzionali a \begin{longtable}

Senza argomenti opzionali, la tabella è sistemata secondo il valore di `\LTleft` e `\LTright`. Quindi, se non sono stati modificati, la tabella è centrata.

```
[c]
```

Centrare la tabella.

```
[l]
```

Tabella al margine sinistro.

```
[r]
```

Tabella al margine destro.

### 11.6 — Comandi per terminare le righe della tabella

```
\
```

Termina una riga.

```
\[/<dimen>]
```

Termina una riga e aggiunge in seguito uno spazio verticale come specificato da `<dimen>` (come in `tabular`).

`\*`

Come `\` ma non permette un cambio pagina dopo la riga.

`\tabularnewline`

Alternativa a `\` da usare quando si sia specificato un comando del tipo di `\raggedright`, `\raggedleft` o `\centering`, che ridefinisce `\`.

`\kill`

La riga è omessa, ma è usata nel calcolo delle dimensioni delle celle.

`\endhead`

Specifica le righe che devono apparire all'inizio di ogni pagina.

`\endfirsthead`

Specifica le righe che appaiono all'inizio della tabella.

`\endfoot`

Specifica le righe che devono apparire alla fine di ogni pagina.

`\endlastfoot`

Specifica le righe che devono apparire alla fine dell'ultima pagina.

### 11.7 — Comandi per la didascalia

`\caption{testo}`

La didascalia sarà 'Tabella *n*: *testo*'; *testo* viene riportato anche nell'elenco delle tabelle. Tipicamente il comando va nell'intestazione iniziale.

`\caption[testo]{testo}`

La didascalia sarà 'Tabella *n*: *testo*'; *testo* viene riportato nell'elenco delle tabelle. Va nell'intestazione iniziale; si usa se il testo della didascalia è troppo lungo.

`\caption[] {testo}`

La didascalia sarà 'Tabella *n*: *testo*'; non sarà riportata nell'elenco delle tabelle; il numero qui rappresentato da '*n*' non cambia rispetto alla prima didascalia. Tipicamente questo comando va nell'intestazione normale.

`\caption*{testo}`

La didascalia sarà '*testo*'; non sarà riportata nell'elenco delle tabelle. Può sostituire il precedente nell'intestazione normale se non si vuole riportare in ogni pagina il numero della tabella.

### 11.8 — Comandi impiegabili all'inizio di una riga

`\pagebreak | \newpage`

Forza un cambio di pagina.

`\pagebreak[numero]`

Specifica un valore fra 0 e 4 di 'desiderabilità' di un cambio pagina.

`\nopagebreak`

Proibisce un cambio di pagina.

`\nopagebreak[<numero>]`

Specifica un valore fra 0 e 4 di ‘non desiderabilità’ di un cambio pagina.

---

**Tabella 11.1** Struttura di una `longtable`. Sono indicate tutte le parti, ma va ricordato che non sono obbligatorie. Non si abusi delle possibilità offerte; per esempio, la didascalia non è sempre necessaria.

---

```
\begin{longtable}{<preambolo>}
<prima riga dell'intestazione iniziale>\
...
<ultima riga dell'intestazione iniziale>\
\caption{<testo>}\endfirsthead
<prima riga dell'intestazione normale>\
...
<ultima riga dell'intestazione normale>\
\caption[]{<testo>}\endhead
<prima riga del piede finale>\
...
<ultima riga del piede finale>\endlastfoot
<prima riga del piede normale>\
...
<ultima riga del piede normale>\endfoot
<prima riga della tabella>\
...
<ultima riga della tabella>
\end{longtable}
```

---

## multicol

`\usepackage[⟨opzione⟩, …]{multicol}`

Scrivere documenti in cui si alternino passi a una colonna e passi a due colonne non è possibile, se non in una forma primitiva: a ogni cambio, si va a pagina nuova. Questo senza usare pacchetti aggiuntivi, ma `multicol` è una cura molto valida: è possibile alternare, nella stessa pagina, tratti a una, a due o a più colonne, con bilanciamento finale. Il numero massimo di colonne è dieci, dovrebbero bastare. Si faccia attenzione che, per ottenere un risultato veramente ottimo, è necessario lavorare parecchio sui parametri di bilanciamento.

Un avvertimento: figure e tabelle inserite in un ambiente `multicols` con `figure*` o `table*` hanno come primo punto possibile di uscita la pagina successiva: è *impossibile* fare in modo che uno di questi ambienti capiti nella *stessa* pagina se non inserendolo *prima o dopo* l'ambiente `multicols`. Se questo non vi va, evitate di usare `multicol`.

### 12.1 — Opzioni

`errorshow` | `infoshow` | `balancingshow` | `markshow` | `debugshow`

Queste opzioni servono per mostrare sul terminale e sul LOG i calcoli e varie altre cose. Utili quando sembra che non si riesca a ottenere il risultato previsto.

`grid`

Le colonne sono composte in modo che le righe corrispondano a quelle di una griglia invisibile. Se non si specifica questa opzione, il bilanciamento delle colonne può creare disparità fra le righe. Attenzione: i parametri di spaziatura dei comandi di sezione e altri devono essere studiati con cura affinché questo possa funzionare.

### 12.2 — Comandi

`\begin{multicols}{⟨numero⟩}[⟨testo⟩][⟨dimen⟩]`

Il `⟨numero⟩` deve essere compreso fra due e dieci (inclusi). Il `⟨testo⟩` dato come argomento opzionale viene composto a larghezza intera, prima del contenuto dell'ambiente, senza che sia possibile un cambio di pagina prima dell'inizio del testo a più colonne. *Gli ambienti `multicols` possono essere annidati*, ma non sempre con risultati ottimali. Il secondo argomento opzionale serve a indicare di quanto spazio abbiamo bisogno perché il testo dell'ambiente e l'intestazione compaiano nella pagina corrente (si veda la discussione sul parametro `\premulticols`).

`\begin{multicols*}{⟨numero⟩}[⟨testo⟩][⟨dimen⟩]`

Il risultato è lo stesso, ma non viene eseguito alcun bilanciamento fra le colonne, che quindi hanno l'altezza massima possibile per riempire la pagina. Non ha senso usare questo ambiente dentro una `minipage`.

`\flushcolumns`

È una dichiarazione valida da quando si carica il pacchetto. Le colonne vengono bilanciate in modo che le linee di base finali di ciascuna colonna (eccetto l'ultima) siano allineate.

`\raggedcolumns`

È il contrario di `\flushcolumns`: le linee di base delle colonne non sono necessariamente allineate.

`\columnbreak`

Indica a  $\text{\LaTeX}$  che la riga in cui compare il comando è l'ultima della colonna (va usato solo in fase di revisione), per ottenere il migliore risultato.

**12.3 — Parametri**

Prima di discuterli, ricordiamo che non è buona pratica giocare con i parametri senza sapere ciò che si fa. E, in ogni caso, gli aggiustamenti vanno fatti quando il testo è in versione definitiva.

`\premulticols`

Questo parametro contiene lo spazio richiesto perché un ambiente `multicols` possa cominciare nella pagina corrente. Se lo spazio rimanente sulla pagina è minore, viene emesso un comando di cambio pagina. Se viene specificato il secondo argomento opzionale a `multicols`, il valore dato è usato al posto di `\premulticols`; usare con cautela.

`\postmulticols`

Se dopo un ambiente `multicols` lo spazio rimanente sulla pagina è minore di questo parametro, viene emesso un comando di cambio pagina.

`\multicolsep`

Spazio verticale che viene inserito prima e dopo un ambiente `multicols` quando non ci siano cambi pagina.

`\columnsep`

Spazio orizzontale fra le colonne. La larghezza delle colonne è determinata da questo parametro e dal valore corrente di `\linewidth`.

`\columnseprule`

Larghezza della linea da inserire per separare le colonne. Il valore usuale è 0 pt. Un valore consigliabile, se si desidera la linea, è 0.4 pt.

`\multicolbaselineskip`

È una lunghezza elastica che viene aggiunta a `\baselineskip` per la composizione del testo in un ambiente `multicols`. Non giocare se non si sa quello che si sta facendo.

`\multicoltolerance`

Valore di `\tolerance` da usare per comporre i paragrafi negli ambienti `multicols`. Il suo valore deve essere cambiato fuori da questi ambienti; usualmente è 9999. Chi non conosce il ruolo di `\tolerance` non ci giochi; può essere modificato localmente assegnando il valore di `\tolerance` all'inizio dell'ambiente.

`\multicolpretolerance`

Valore di `\pretolerance` da usare per comporre i paragrafi negli ambienti `multicols`. Il suo valore deve essere cambiato fuori da questi ambienti; usualmente è  $-1$ . Chi non conosce il ruolo di `\pretolerance` non ci giochi può essere modificato localmente assegnando il valore di `\pretolerance` all'inizio dell'ambiente.

`collectmore`

L'intero contenuto in questo contatore (usualmente 0) indica quante righe di testo vanno considerate in più o in meno per ogni colonna, eccetto l'ultima, rispetto a quanto sarebbe considerato dalla procedura di impaginazione.

***unbalance***

Questo contatore deve avere un valore non negativo. Quando ha valore positivo, la procedura di impaginazione crea colonne con un numero di righe superiore al normale proprio del valore di questo contatore. Viene riportato a zero alla fine di ogni ambiente `multicols`.

***columnbadness***

La procedura di impaginazione scarta ogni soluzione in cui le colonne (eccetto l'ultima) abbiano un valore di `\vbadness` che ecceda il valore di questo contatore. Il valore usuale è 10000, cioè la procedura accetta tutte le soluzioni tranne quelle in cui la colonna sia *overflow*. Questo può portare a colonne con ampi spazi bianchi; si può, in fase di revisione, provare a ridurre questo valore. Valori troppo bassi potrebbero forzare la procedura a mettere tutto il testo nella prima colonna.

***finalcolumnbadness***

Se la `\vbadness` dell'ultima colonna è minore del valore di questo contatore, usualmente 9999, l'ultima colonna stessa viene allungata in modo da essere bilanciata con le altre.

**showkeys**

```
\usepackage[⟨opzione⟩, …]{showkeys}
```

Permette di vedere le etichette assegnate tramite `\label` o `\bibitem` e quelle richiamate con `\ref` o `\cite`. È quindi molto utile quando si stia preparando un documento con molti riferimenti incrociati.

**13.1 — Opzioni**

`notcite`

Non vengono mostrate le etichette richiamate con `\cite`.

`notref`

Non vengono mostrate le etichette richiamate con `\ref`.

`draft`

È il comportamento usuale del pacchetto: mostra le etichette.

`final`

Nessuna etichetta viene mostrata. Si consiglia però di eliminare il caricamento del pacchetto quando si prepara la versione finale del documento, perché in alcuni casi la sua presenza può influenzare l'impaginazione.

`color`

Colora le etichette. Il colore usuale è un grigio leggero, che può essere modificato assegnando significati diversi ai colori `refkey` e `labelkey`. Con questa opzione viene caricato il pacchetto `color`. Un modo di ridefinire i colori può essere

```
\definecolor{labelkey}{rgb}{1,0,0}
```

che mostrerà le etichette in rosso.



### tabularx

```
\usepackage[infoshow,debugshow]{tabularx}
```

L<sup>A</sup>T<sub>E</sub>X mette a disposizione l'ambiente `tabular*` per costruire tabelle di una fissata larghezza. Tuttavia lo fa aumentando lo spazio fra le colonne. Il pacchetto `tabularx` invece aumenta la larghezza di certe colonne indicate con il nuovo specificatore `X`. Il pacchetto carica `array`.

Le due opzioni possibili sono equivalenti e mostrano al terminale e nel LOG i calcoli eseguiti per stabilire le dimensioni delle colonne.

```
\begin{tabularx}{\langle dimen \rangle}{\langle preambolo \rangle}
```

Comincia una tabella la cui larghezza è  $\langle dimen \rangle$ ; le colonne specificate con gli usuali caratteri (`l`, `c`, `r`, `p`, `m`, `b`) avranno la loro larghezza naturale. Si sottrae a  $\langle dimen \rangle$  la somma di queste larghezze e la somma degli spazi tra colonne; la larghezza che rimane viene divisa equamente tra le colonne specificate con `X`.

**X**

Una colonna specificata così è analoga a una colonna di tipo `p`, ma non si deve assegnarle una larghezza che invece verrà calcolata automaticamente.

```
\tracingtabularx
```

È una dichiarazione equivalente alle opzioni date al pacchetto. Utile se si vuole capire che cosa non va in una certa tabella quando non sembra fare ciò che desideriamo: si apre un gruppo, si dà il comando, si compone la tabella e si chiude il gruppo.

## varioref

```
\usepackage[⟨lingua⟩ | draft | final]{varioref}
```

Quando il riferimento a una figura o una tabella fuori testo appare in una pagina diversa da dove sono quegli oggetti, è utile indicare anche il numero di pagina. La possibile soluzione

```
\newcommand{\completeref}[1]{\ref{#1} a pagina~\pageref{#1}}
```

non è soddisfacente perché, dopo tutto, la figura potrebbe capitare proprio nella stessa pagina e si dovrebbe modificare il testo del documento per tenerne conto. Il pacchetto `varioref` cerca di risolvere il problema.

Attenzione: ogni uso dei comandi `\vref` e `\vpageref` genera due comandi interni; ciò può provocare problemi con la memoria di  $\TeX$ . Si usi il comando `\fullref` se si è sicuri che l'oggetto riferito è 'lontano'.

### 15.1 — Opzioni

```
⟨lingua⟩
```

Una delle lingue accettate da `babel`. Non per tutte il supporto è completo. Per l'italiano lo è quasi.

```
draft | final
```

La prima è l'opzione usuale. Eventuali situazioni a rischio vengono segnalate con un avvertimento. Con l'opzione `final` queste situazioni producono un errore.

Le situazioni a rischio sono rare, ma possono capitare. Se il riferimento a una figura capita vicino a un cambio pagina, aggiungere al numero della figura le parole 'nella pagina seguente' potrebbe far scattare il cambio di pagina e il riferimento potrebbe cadere sulla *stessa pagina* della figura! La soluzione per questi casi particolari deve essere trovata volta per volta.

Il pacchetto cerca anche di evitare ripetizioni della stessa formula. Si cerchi di non abusare delle possibilità offerte, impiegandole solo quando è veramente necessario per la chiarezza.

### 15.2 — Comandi

```
\fullref{⟨etichetta⟩}
```

Produce il numero a cui si riferisce l'etichetta seguito da "a pagina  $\langle num \rangle$ ", dove  $\langle num \rangle$  è il numero di pagina dove compare l'oggetto a cui ci si riferisce.

```
\vref{⟨etichetta⟩}
```

Se l'oggetto a cui ci si riferisce cade nella stessa pagina, il comando è equivalente a `\ref`. Altrimenti diventa come

```
\ref{⟨etichetta⟩} nella pagina precedente\\
\ref{⟨etichetta⟩} nella pagina a fronte\\
\ref{⟨etichetta⟩} nella pagina successiva\\
\ref{⟨etichetta⟩} a pagina~\pageref{⟨etichetta⟩}
```

a seconda dei casi. La locuzione 'a fronte' viene usata se la classe del documento usa l'opzione `twoside` e l'oggetto riferito cade sulla stessa coppia di pagine affacciate rispetto al richiamo. Se oggetto e riferimento cadono su pagine che differiscono di più di uno, viene

usato il quarto modo. Lo stesso accade in ogni caso quando la numerazione delle pagine non è con numeri arabi.

```
\vpageref[⟨testo1⟩][⟨testo2⟩]{⟨etichetta⟩}
```

Se l'oggetto a cui ci si riferisce cade nella stessa pagina, viene prodotto il testo contenuto nel comando `\ref`. Altrimenti si comporta in modo analogo a `\vref`. Per capire l'uso, compreso quello dei due argomenti opzionali, facciamo un esempio.

```
... si veda l'esempio \vpageref{es:pippo} che mostra ...
```

Verrà prodotto il testo seguente, a seconda dei casi.

```
... si veda l'esempio in questa pagina che mostra ...
... si veda l'esempio nella pagina precedente che mostra ...
... si veda l'esempio nella pagina a fronte che mostra ...
... si veda l'esempio nella pagina successiva che mostra ...
... si veda l'esempio a pagina xx che mostra ...
```

Possiamo non essere soddisfatti del testo nel primo caso, magari preferiamo dire 'l'esempio seguente' se l'esempio cade nella stessa pagina.

```
... si veda l'esempio \vpageref[seguente]{es:pippo} che mostra ...
```

Questo produrrà il testo che mostriamo, a seconda dei casi (manca, ovviamente, il caso in cui l'oggetto riferito viene prima).

```
... si veda l'esempio seguente che mostra ...
... si veda l'esempio nella pagina a fronte che mostra ...
... si veda l'esempio nella pagina successiva che mostra ...
... si veda l'esempio a pagina xx che mostra ...
```

Potremmo però non gradire 'l'esempio seguente' e preferire 'il seguente esempio'.

```
... si veda \vpageref[il seguente esempio][l'esempio]{es:pippo}
che mostra ...
```

Questo produce uno dei testi che mostriamo.

```
... si veda il seguente esempio che mostra ...
... si veda l'esempio nella pagina a fronte che mostra ...
... si veda l'esempio nella pagina successiva che mostra ...
... si veda l'esempio a pagina xx che mostra ...
```

```
\vrefrange{⟨etichetta1⟩}{⟨etichetta2⟩}
```

Se vogliamo riferirci a una serie di figure, o di risultati, o di equazioni, o di tabelle, comunque di oggetti a cui abbiamo assegnato un'etichetta, possiamo dare questo comando che genererà i riferimenti al primo (quello contrassegnato con  $\langle etichetta_1 \rangle$ ) all'ultimo (quello contrassegnato con  $\langle etichetta_2 \rangle$ ) separati da un trattino medio e aggiungerà anche i riferimenti alla serie di pagine in cui compaiono (ne scriverà una sola se è il caso, con il sistema visto per `\vref`). Per esempio,

```
... si vedano le figure \vrefrange{fig:pippo}{fig:pluto} ...
```

potrebbe generare

```
... si vedano le figure da 3.2 a 3.4 alle pagine 24–26 ...
```

oppure

```
... si vedano le figure da 3.2 a 3.4 a pagina 24 ...
```

oppure ancora

```
... si vedano le figure da 3.2 a 3.4 nella pagina precedente ...
```

o altre variazioni.

```
\vpagerefrange[⟨testo⟩]{⟨etichetta1⟩}{⟨etichetta2⟩}
```

È analogo al precedente, si confrontino le differenze fra `\vref` e `\vpageref`. L'argomento opzionale serve per il testo da comporre nel caso gli oggetti riferiti appaiano nella stessa pagina in cui c'è il richiamo tramite il comando.

```
\vrefpagenum{⟨comando⟩}{⟨etichetta⟩}
```

Il primo argomento è un nome di comando (con la barra rovescia), il secondo un'etichetta. L'effetto è di definire il `⟨comando⟩` come il numero di pagina dove compare l'oggetto contrassegnato da `⟨etichetta⟩`. Attenzione: non c'è alcun controllo se il `⟨comando⟩` sia già definito, usare con cautela. Vediamo un esempio d'uso (che impiega `ifthen`).

```
\newcommand{\divertimento}[2]{%
  \vrefpagenum{\firstnum}{#1}%
  \vrefpagenum{\secondnum}{#2}%
  \ifthenelse{\equal{\firstnum}\secondnum}%
    {le equazioni \ref{#1} e \ref{#2} \vpageref{#1}}%
    {l'equazione \ref{#1} \vpageref{#1}%
     e l'equazione \ref{#2} \vpageref{#2}}%
}
```

nel preambolo e

```
... \divertimento{pippo}{pluto}
```

potrebbe dare

```
... le equazioni 3 e 4 nella pagina precedente
```

oppure

```
... l'equazione 3 a pagina 24 e l'equazione 4 alla pagina a fronte
```

```
\vref* | \vpageref* | \vpagerefrange*
```

Per un errore di progettazione, i comandi con lo stesso nome ma senza asterisco producono automaticamente uno spazio indivisibile alla loro sinistra. Ciò li rende inutili se vogliamo, per esempio, un riferimento tra parentesi. In questo caso si deve usare la variante con l'asterisco.

### 15.3 — I comandi che producono i testi variabili

Elenchiamo i comandi che è possibile modificare, indicando la loro definizione usuale in italiano. Per ridefinirli, occorre usare

```
\addto{extrasitalian}{
  \renewcommand{⟨comando⟩}[⟨numero⟩]{⟨definizione⟩}
  ...
}
```

dove `⟨comando⟩` è il nome del comando, `⟨numero⟩` il numero di argomenti e `⟨definizione⟩` il testo della definizione. Indicheremo solo il numero di argomenti e la `⟨definizione⟩` usuale. Si usi lo stesso schema.

```
\reftextfaceafter
```

Contiene il testo per i riferimenti alla pagina a fronte successiva. Non ha argomenti.

```
\reftextvario{a fronte}{nella pagina successiva}
```

`\reftextfacebefore`

Contiene il testo per i riferimenti alla pagina a fronte precedente. Non ha argomenti.

`\reftextvario{a fronte}{nella pagina precedente}`

`\reftextafter`

Contiene il testo per i riferimenti alla pagina successiva. Non ha argomenti.

nella pagina `\reftextvario{seguinte}{successiva}`

`\reftextbefore`

Contiene il testo per i riferimenti alla pagina precedente. Non ha argomenti.

nella pagina precedente

`\reftextcurrent`

Contiene il testo per i riferimenti alla stessa pagina. Non ha argomenti.

in questa pagina

`\reftextfaraway`

Contiene il testo per i riferimenti a una pagina lontana. Ha un argomento.

a pagina `\pageref{#1}`

`\reftextpagerange`

Questo comando ha due argomenti. Non è ben definito nella versione italiana. Si consiglia di definirlo (se si usano `\vrefrange` o `\vpagerefrange`) usando, come *<definizione>*

alle pagine `\pageref{#1}--\pageref{#2}`

`\reftextlabelrange`

Questo comando ha due argomenti. Non è ben definito nella versione italiana. Si consiglia di definirlo (se si usano `\vrefrange` o `\vpagerefrange`) usando, come *<definizione>*

da `\ref{#1}` a `\ref{#2}`

**verbatim**

```
\usepackage{verbatim}
```

L'ambiente `verbatim` fornito dal nucleo di  $\text{\LaTeX}$  ha certi difetti che lo rendono poco flessibile: per esempio è complicato ottenere che il testo da rendere *verbatim* (latino medievale, significa 'parola per parola') sia in corpo minore.

Un difetto maggiore è che il contenuto di ciascun ambiente `verbatim` deve essere raccolto completamente in memoria prima di venire elaborato; ciò può facilmente provocare problemi.

**16.1 — Ambienti**

```
\begin{verbatim} | \begin{verbatim*}
```

Il testo compreso fra questo comando e il corrispondente `\end{verbatim}` viene reso così com'è, senza interpretare alcun carattere speciale, compresa la barra rovescia. Si faccia attenzione che, a differenza dell'ambiente fornito dal nucleo di  $\text{\LaTeX}$ , ciò che segue `\end{verbatim}` sulla stessa riga del documento `.tex` viene *scartato* (con messaggio di avviso). Fra `\end` e `{verbatim}` può esserci uno spazio, ma non un a capo. La variante `*` è analoga, ma gli spazi vengono stampati come `␣`.

```
\begin{comment}
```

Il testo compreso fra questo comando e il corrispondente `\end{comment}` viene *ignorato*. Ha le stesse limitazioni di `verbatim` per quanto riguarda testo che compaia dopo `\end{comment}` sulla stessa riga. È utile per commentare intere sezioni di testo che non vogliamo far comparire nel documento finale, ma che vogliamo mantenere nel sorgente.

```
\verbatiminput{<nome>}
```

Il contenuto del *file* `<nome>` viene inserito come se si trovasse all'interno di un ambiente `verbatim`, cioè senza che siano interpretati i caratteri speciali. Può servire, per esempio, per citare il codice di un programma.

**16.2 — Creare nuovi ambienti di tipo `verbatim`**

```
\verbatim, \endverbatim
```

Questi comandi possono essere impiegati per definire ambienti personalizzati. Si supponga, per esempio, di voler citare alcune righe di codice in corpo minore, in modo che queste righe comincino a una distanza fissata dal margine sinistro, diciamo il doppio del rientro usuale.

```
\newenvironment{codice}
  {\par\medskip
   \noindent\hspace{2\parindent}%
   \minipage{\textwidth-2\parindent}
   \small\verbatim}
  {\endverbatim\endminipage\par\medskip}
```

Si noti che abbiamo usato la sintassi permessa da `calc`. Questo non è obbligatorio; obbligatorio è usare comandi e non ambienti all'interno di queste definizioni. Per esempio l'ambiente `minipage` è sostituito da `\minipage` e `\endminipage`. Si noti come l'ultimo comando della prima parte sia `\verbatim` e il primo della seconda parte sia `\endverbatim`.

## caption

```
\usepackage[⟨variabile⟩=⟨valore⟩,...]{caption}[2004/07/16]
```

Uno dei punti tipograficamente meno validi degli stili standard di L<sup>A</sup>T<sub>E</sub>X è quello delle didascalie a figure o tabelle fuori testo. Con il pacchetto `caption` è possibile modificarne l'aspetto. Il pacchetto è anche compatibile con `float`, che permette di definire nuovi oggetti 'flottanti'.

Attenzione: una vecchia versione di questo pacchetto potrebbe essere ancora in giro, se la distribuzione che usate non è aggiornata. Per evitare di trovarsi errori misteriosi, è meglio chiamare questo pacchetto con l'argomento opzionale indicato come ultimo; solo `caption` dalla versione 3 in poi sarà ammesso.

La didascalia a pagina 28 è composta in corpo minore rispetto al testo normale, con l'etichetta in nero e margini laterali uguali al rientro dei paragrafi. È stato usato `caption`.

Una didascalia è formata da tre parti:

- un'etichetta, che dice se stiamo commentando una figura o una tabella e ne riporta il numero;
- il corpo, cioè il testo del commento;
- lo spazio di separazione con il testo del documento.

### 17.1 — Opzioni

Le opzioni al pacchetto vanno date nella forma `⟨variabile⟩=⟨valore⟩`. Se dopo il segno di uguale compaiono stringhe tra parentesi graffe, quelle sono i valori ammissibili fra cui scegliere. Negli esempi i filetti verticali rappresentano i margini della gabbia, in modo da mostrare l'effetto.

```
format={default,hang}
```

Se non si specifica l'opzione, il valore è `default`. Dando il valore `hang`, il testo della didascalia verrà allineato a sinistra con il margine destro dell'etichetta.

- `format=default`

Figura 17.1: Notte e giorno faticar, per chi nulla sa gradir, piova e vento sopportar, mangiar male, mal dormir! Voglio fare il gentiluomo e non voglio più servir! Ah, che caro galantuomo, vuol star dentro con la bella e io a far la sentinella!

- `format=hang`

Figura 17.2: Notte e giorno faticar, per chi nulla sa gradir, piova e vento sopportar, mangiar male, mal dormir! Voglio fare il gentiluomo e non voglio più servir! Ah, che caro galantuomo, vuol star dentro con la bella e io a far la sentinella!

```
indentation=⟨dimen⟩
```

Aggiunge un rientro di `⟨dimen⟩` dalla seconda riga della didascalia. Può essere anche negativo.

- `format=default,indentation=2em`

Figura 17.3: Notte e giorno faticar, per chi nulla sa gradir, piova e vento sopportar, mangiar male, mal dormir! Voglio fare il gentiluomo e non voglio più servir! Ah, che caro galantuomo, vuol star dentro con la bella e io a far la sentinella!

- `format=hang,indentation=-2em`

Figura 17.4: Notte e giorno faticar, per chi nulla sa gradir, piova e vento sopportar, mangiar male, mal dormir! Voglio fare il gentiluomo e non voglio più servir! Ah, che caro galantuomo, vuol star dentro con la bella e io a far la sentinella!

`labelformat={empty, simple, parens}`

Con `empty` non ci sarà alcuna etichetta; non ha senso usato da solo, si vedano le altre variabili come `labelsep`. Con `simple` si ha l'usuale "Figura *n*". Con `parens`, il numero della figura viene messo fra parentesi.

- `labelformat=parens`

Figura (17.5): Notte e giorno faticar, per chi nulla sa gradir, piova e vento sopportar, mangiar male, mal dormir! Voglio fare il gentiluomo e non voglio più servir! Ah, che caro galantuomo, vuol star dentro con la bella e io a far la sentinella!

`labelsep={none, colon, period, space, quad, newline}`

Si specifica come separare l'etichetta dal testo; con `none` non si ha nulla (ha senso con `labelformat=empty`); con `colon` si hanno gli usuali due punti, con `period` un punto, con `space` un normale spazio fra parole, con `quad` uno spazio quadratone. Con `newline` il testo della didascalia comincia una riga sotto; in tal caso meglio usare l'opzione `singlelinecheck=false`.

- `labelsep=newline`

Figura 17.6  
Notte e giorno faticar, per chi nulla sa gradir, piova e vento sopportar, mangiar male, mal dormir! Voglio fare il gentiluomo e non voglio più servir! Ah, che caro galantuomo, vuol star dentro con la bella e io a far la sentinella!

`justification={justified, centerlast}`

Con `justified` il testo è composto giustificato (è l'opzione usuale). Con `centerlast` l'ultima riga di testo è centrata.

Esistono anche i valori `centerfirst`, `raggedright`, `RaggedRight` e `raggedleft` per i quali rimandiamo alla documentazione.

- `justification=centerlast`

Figura 17.7: Notte e giorno faticar, per chi nulla sa gradir, piova e vento sopportar, mangiar male, mal dormir! Voglio fare il gentiluomo e non voglio più servir! Ah, che caro galantuomo, vuol star dentro con la bella e io a far la sentinella!

`singlelinecheck={true, false}`

Usuale è il valore `true`. Se la variabile ha questo valore, una didascalia breve viene centrata, indipendentemente dal valore di `justification`. Sinonimi per `true` sono `yes` e `1`. Sinonimi per `false` sono `no` e `0`.

- `singlelinecheck=false`

Figura 17.8: Didascalia breve

- `singlelinecheck=true`

Figura 17.9: Didascalia breve



`font=<insieme di valori>`

Questa variabile ha come valore un insieme. Si possono specificare la grandezza e la forma del carattere da usare per la didascalia. Il suo valore influenza sia l'etichetta che il testo. I valori per la grandezza sono nell'insieme

`{scriptsize,footnotesize,small,normalsize,large,Large}`

che si spiegano da soli. I valori per la forma sono nell'insieme

`{up,it,sl,sc,md,bf,rm,sf,tt}`

Con `up` si specifica un carattere diritto; con `it` un carattere corsivo; con `sl` un carattere inclinato; con `sc` un carattere maiuscoletto. Con `md` si chiede un peso medio, con `bf` il neretto. Con `rm` si sceglie il carattere usuale, con `sf` uno senza grazie, con `tt` quello a spaziatura fissa.

È evidente che non tutte queste opzioni hanno senso. È opportuno scegliere, per esempio, `small`, ed è possibile chiedere il corsivo.

- `font={small,it}`

*Figura 17.10: Notte e giorno faticar, per chi nulla sa gradir, piova e vento sopportar, mangiar male, mal dormir! Voglio fare il gentiluomo e non voglio più servir! Ah, che caro galantuomo, vuol star dentro con la bella e io a far la sentinella!*

`labelfont=<insieme di valori>`

Vale lo stesso discorso di prima, ma il valore di questa variabile influenza solo l'etichetta.

`textfont=<insieme di valori>`

Vale lo stesso discorso di prima, ma il valore di questa variabile influenza solo il testo.

Si userà quindi `font` per fissare le caratteristiche comuni, le altre due per quelle specifiche.

- `font=small,labelfont=bf,textfont=it`

**Figura 17.11:** *Notte e giorno faticar, per chi nulla sa gradir, piova e vento sopportar, mangiar male, mal dormir! Voglio fare il gentiluomo e non voglio più servir! Ah, che caro galantuomo, vuol star dentro con la bella e io a far la sentinella!*

- `font={small,it},labelfont=bf`

**Figura 17.12:** *Notte e giorno faticar, per chi nulla sa gradir, piova e vento sopportar, mangiar male, mal dormir! Voglio fare il gentiluomo e non voglio più servir! Ah, che caro galantuomo, vuol star dentro con la bella e io a far la sentinella!*

`margin=<dimen>`

Specifica la larghezza del margine bianco a sinistra e a destra della didascalia.

- `margin=2pc,font=small,labelfont=bf`

**Figura 17.13:** *Notte e giorno faticar, per chi nulla sa gradir, piova e vento sopportar, mangiar male, mal dormir! Voglio fare il gentiluomo e non voglio più servir! Ah, che caro galantuomo, vuol star dentro con la bella e io a far la sentinella!*

`width=<dimen>`

Specifica la larghezza della riga per la didascalia.

- `width=24pc`

Figura 17.14: Notte e giorno faticar, per chi nulla sa gradir, piova e vento sopportar, mangiar male, mal dormir! Voglio fare il gentiluomo e non voglio più servir! Ah, che caro galantuomo, vuol star dentro con la bella e io a far la sentinella!

`\parskip=<dimen>`

Specifica la distanza fra due paragrafi nella didascalia. Si consiglia, però, di non essere prolissi.

`\hangindent=<dimen>`

Simile a `\indention`, più utile se la didascalia ha più di un paragrafo.

`\style=default`

È possibile definire stili di didascalie in modo analogo agli stili di pagina. Il pacchetto definisce solo quello indicato.

`\aboveskip=<dimen>`

Specifica la spaziatura sopra la didascalia. Il valore usuale per le classi di L<sup>A</sup>T<sub>E</sub>X è 10 pt.

`\belowskip=<dimen>`

Specifica la spaziatura sotto la didascalia. Il valore usuale per le classi di L<sup>A</sup>T<sub>E</sub>X è 0 pt.

`\position={top,bottom}`

I parametri di spaziatura vengono scambiati se si usa il valore `top`.

`\tableposition=top`

Va data come opzione se si preferisce, come da certe usanze tipografiche, porre la didascalia delle tabelle in testa.

## 17.2 — Comandi

`\caption[<testo1>]{<testo2>}`

È ridefinito rispetto a quello del nucleo di L<sup>A</sup>T<sub>E</sub>X. Un argomento opzionale vuoto evita che la tabella o figura compaia nella lista relativa.

`\caption*{<testo>}`

Il `<testo>` forma la didascalia, senza etichetta; la figura o tabella non comparirà nella lista relativa. In questo caso non ha senso dare il comando `\label`.

`\captionof{<tipo>}[<testo1>]{<testo2>}`

Compone la didascalia come se si fosse all'interno dell'ambiente `<tipo>` (cioè `figure` o `table`, oppure un ambiente definito tramite il pacchetto `float`). Utile se si vuole simulare un oggetto 'flottante' con una `minipage` per evitare che si sposti dal punto ideale. Tuttavia le figure o tabelle potrebbero risultare fuori ordine, usatelo con cautela.

`\captionof*{<tipo>}{<testo>}`

Simile al precedente, ma non assegna un numero, come `\caption*`.

`\ContinuedFloat`

Se una tabella deve essere spezzata in più parti, questo comando diventa molto utile. Dato subito dopo `\begin{table}`, fa in modo che il contatore non sia aumentato dal comando

`\caption`. Sarà cura dell'utente scrivere una didascalia che indichi come la tabella sia una prosecuzione della precedente.

```
\captionsetup[<tipo>]{<opzioni>}
```

L'argomento obbligatorio è scritto come quello opzionale del pacchetto. Per esempio,

```
\usepackage[margin=2pc,font=small]{caption}
```

è equivalente a

```
\usepackage{caption}
\captionsetup{margin=2pc,font=small}
```

ma il comando ha un uso più importante. L'argomento opzionale contiene un *<tipo>* di oggetto 'flottante' (**figure** o **table**), ciò che permette di definire stili diversi per le didascalie dei vari oggetti. Non che sia consigliabile, naturalmente, in generale; tuttavia possiamo dire esplicitamente che vogliamo avere le didascalie per le tabelle in alto e quelle per le figure in basso:

```
\captionsetup[table]{position=top}
\captionsetup[figure]{position=bottom}
```

Il pacchetto usa di suo una regola 'euristica' per decidere la spaziatura fra tabella o figura e didascalia; in questo modo eliminiamo il rischio di errori. Ovviamente dovremo scrivere noi la didascalia al posto giusto con il comando `\caption`.

Il comando può anche essere usato nel documento, oltre che nel preambolo, per cambiare lo stile delle didascalie. Può essere indicato per le figure o tabelle fuori testo in un'appendice, per esempio.

```
\clearcaptionsetup{<tipo>}
```

Azzerare tutte le personalizzazioni date alle didascalie dell'ambiente *<tipo>*, riportandole ai valori definiti dalla classe.

### 17.3 — Altri pacchetti

Il pacchetto `caption` è compatibile con `float`, `listings`, `longtable`, `rotating`, `sidecap` e `supertabular`. Come già detto, dopo aver dato il comando `\newfloat{pippo}` di `float`, è disponibile

```
\captionsetup[pippo]{<opzioni>}
```

e così è disponibile anche

```
\captionsetup[lstlisting]{<opzioni>}
```

se si usa `listings`. Tuttavia non si può usare in questo modo l'ambiente `sideways` di `rotating`.

## float

```
\usepackage{float}
```

Gli ambienti `figure` e `table` non sempre sono sufficienti alle esigenze di chi scrive. Un documento potrebbe avere bisogno anche di ‘algoritmi’, ‘grafici’ o altro. Il pacchetto `float` risponde a questa esigenza e permette anche di modificare il modo con cui gli ambienti tradizionali vengono composti, introducendo il concetto di *stile* per questi oggetti ‘galleggianti’.

## 18.1 — Stili

Gli stili predefiniti sono:

**plain** che corrisponde allo stile usuale delle classi di L<sup>A</sup>T<sub>E</sub>X;

**plaintop** che è simile al precedente, ma fa sì che la didascalia sia sempre sopra all’oggetto;

**boxed** che racchiude l’oggetto in un riquadro;

**ruled** che pone la didascalia in alto, separata dal testo precedente e dall’oggetto da una linea; un’altra linea separa l’oggetto dal testo seguente (l’esempio è quello di pagina 28).

## 18.2 — Comandi

```
\newfloat{<nome>}{<pos>}{<ext>}[<contatore>]
```

Definisce un nuovo tipo di oggetto ‘galleggiante’ e il relativo ambiente  $\langle nome \rangle$ . Il secondo argomento specifica i valori di preferenza per il posizionamento dell’oggetto (di solito si dà `htp`) che saranno usati in mancanza di indicazioni esplicite. Il terzo argomento specifica l’estensione per il *file* ausiliario che serve per compilare la lista, analogamente a `lof` e `lot` per le liste di figure e tabelle. L’argomento opzionale serve per stabilire rispetto a quale contatore devono essere numerati questi oggetti; per esempio, nella classe `book` le figure e tabelle vengono numerate secondo il contatore `chapter`.

```
\floatstyle{<stile>}
```

Definisce quale stile (vedi sopra) viene usato per i nuovi oggetti introdotti con `\newfloat`. È una dichiarazione *globale*.

```
\floatname{<nome>}{<testo>}
```

Dichiara che l’oggetto ‘galleggiante’  $\langle nome \rangle$  sia etichettato con  $\langle testo \rangle$ .

```
\floatstyle{ruled}
\newfloat{algorithm}{htp}{loa}[chapter]
\floatname{algorithm}{Algoritmo}
```

```
\floatplacement{<nome>}{<pos>}
```

Dichiara un nuovo ordine di preferenza per il piazzamento degli oggetti  $\langle nome \rangle$ , ancora in termini di `h`, `t`, `p` oppure `b`.

```
\restylefloat{<nome>}
```

Ridisegna l’oggetto ‘galleggiante’ secondo lo stile valido in quel momento (e dichiarato con `\floatstyle`). È bene che gli oggetti siano trattati in modo uniforme, così è possibile scrivere

```
\floatstyle{ruled}
\restylefloat{figure}
```

cioè si possono applicare i nuovi stili anche a oggetti definiti dal nucleo di L<sup>A</sup>T<sub>E</sub>X.

```
\listof{<nome>}{<testo>}
```

Compila in quel punto la lista degli oggetti *<nome>* con *<testo>* come titolo, per esempio

```
\listof{algorithm}{Elenco degli algoritmi}
```

Può essere usato anche con gli ambienti `figure` e `table`.

```
\floatevery{<nome>}{<testo>}
```

Il *<testo>* dovrebbe essere una sequenza di comandi da applicare a ciascun ambiente *<nome>*. Per esempio,

```
\floatevery{figure}{\centering}
```

dopo aver dato `\restylefloat{figure}` permette di non dover scrivere `\centering` in ogni ambiente `figure`.

### 18.3 — L'argomento di posizione H

Una delle PPF (*proteste più frequenti*) è: 'L<sup>A</sup>T<sub>E</sub>X non mette le figure dove voglio io; non voglio che se ne vadano in giro!'

Il pacchetto `float` mette a disposizione, come argomento opzionale degli ambienti 'galleggianti', anche H che sta per '*proprio qui e in nessun altro posto*'. Questo non può essere usato con altri caratteri come `h`, `t`, `p` oppure `b` (ovviamente) né nell'argomento *<pos>* di `\newfloat`.

Se la resa tipografica di un documento nel quale impiegate questo trucco non è ottima, non prendetevela né con l'autore del pacchetto né con l'autore di questa breve guida. Ci sono buonissime ragioni perché le figure vadano 'a spasso'; se non fossero figure fuori testo, avrebbero infatti il loro posto.

**subfig**

```
\usepackage[⟨opzione⟩,...]{subfig}
```

Il pacchetto `subfig` è il successore del noto `subfigure`, scritto dallo stesso autore. Il suo scopo è di permettere di affiancare più figure o tabelle, dando a ciascuna una *sottodidascalia*. Richiede la presenza del pacchetto `caption`, sul quale si appoggia pesantemente.

**19.1 — Opzioni**

Il pacchetto `subfig` accetta tutte le opzioni di `caption`, ma le usa per la composizione delle sottodidascalie. Per evitare di dover scrivere tutte le opzioni come argomento opzionale in `\usepackage{subfig}`, è possibile usare la sintassi di `caption` secondo lo schema

```
\captionsetup[subfloat]{⟨opzioni⟩}
```

per modificare la forma di composizione delle sottodidascalie. Rimando al capitolo 17 per le opzioni fornite dal pacchetto `caption` e discuterò solo quelle specifiche di `subfig`. Va tenuto presente che il formato della numerazione della sottofigura o sottotabella è, se non modificato, alfabetico e fra parentesi, quindi del tipo “(a)”, “(b)” e così via. I contatori che vengono impegnati sono *subfigure* e *subtable*.

Le opzioni sono del tipo  $\langle chiave \rangle = \langle valore \rangle$ .

```
listofformat=
```

Serve per stabilire il formato delle didascalie riportate nell’elenco delle tabelle o delle figure. I valori sono descritti di seguito:

`empty` per non avere né il numero né il tioletto, ma solo la sottodidascalia;

`simple` per avere il tioletto seguito dal numero della sottofigura e dalla sottodidascalia;

`parens` è come il precedente ma il numero è fra parentesi;

`subsimple` per avere solo il numero della sottofigura e la sottodidascalia;

`subparens` è come il precedente, ma il numero è fra parentesi.

```
listofindent=
```

Prende come valore una dimensione che specifica il rientro il rientro dal margine sinistro delle sottodidascalie nell’elenco delle tabelle o delle figure.

```
listofnumwidth=
```

Prende come valore una dimensione, lo spazio riservato al numero delle sottofigure nell’elenco delle tabelle o delle figure.

Per capire le prossime opzioni, facciamo riferimento alle due situazioni possibili: sottodidascalia in basso o in alto rispetto alla sottofigura o sottotabella, si veda la Tabella 19.1.

```
farskip=
```

Prende come argomento una “lunghezza elastica”, indica lo spazio lasciato fra la sottofigura e l’ambiente circostante dalla parte opposta rispetto alla didascalia.

**Tabella 19.1** Nella colonna di sinistra è rappresentata la sequenza di oggetti quando la sottodidascalia è in basso rispetto alla sottofigura o sottotabella, in quella di destra la situazione opposta. Qui  $\langle farskip \rangle$ ,  $\langle captionskip \rangle$ ,  $\langle nearskip \rangle$  e  $\langle topadjust \rangle$  sono spazi verticali, mentre  $\langle subfloat \rangle$  e  $\langle subcaption \rangle$  rappresentano la sottofigura o sottotabella e la sottodidascalia rispettivamente.

$\langle farskip \rangle$	$\langle nearskip \rangle$
$\langle subfloat \rangle$	$\langle subcaption \rangle$
$\langle captionskip \rangle$	$\langle captionskip \rangle$
$\langle subcaption \rangle$	$\langle topadjust \rangle$
$\langle nearskip \rangle$	$\langle subfloat \rangle$
	$\langle farskip \rangle$

`nearskip=`

Prende come argomento una “lunghezza elastica”, indica lo spazio lasciato fra la sottofigura e l’ambiente circostante dalla parte della didascalia.

`captionskip=`

Prende come argomento una “lunghezza elastica”, indica lo spazio lasciato fra la sottofigura e la didascalia.

`topadjust=`

Prende come argomento una “lunghezza elastica”, indica uno spazio supplementare lasciato fra la sottofigura e la didascalia quando questa è in alto.

## 19.2 — Comandi

`\subfloat [didascalia_breve] [didascalia] {sottooggetto}`

Inserisce una sottofigura o sottotabella, che qui viene chiamata *sottooggetto*, un oggetto grafico o un ambiente `tabular`, per esempio. I due argomenti opzionali hanno lo stesso ruolo dell’argomento opzionale e obbligatorio del comando `\caption`: il primo, se presente, indica ciò che va nell’elenco relativo (`\listoffigures` o `\listoftables`), mentre il secondo indica la sottodidascalia da porre nell’ambiente ‘galleggiante’. Tuttavia se sono presenti e vuoti oppure assenti, indicano precisi comportamenti a L<sup>A</sup>T<sub>E</sub>X. I casi possibili sono quindi sette:

- `\subfloat {sottooggetto}`: il sottooggetto non ha sottodidascalia né numero;
- `\subfloat [] {sottooggetto}`: il sottooggetto ha una sottodidascalia che consiste solo del numero; questo compare anche nell’elenco relativo;
- `\subfloat [didascalia] {sottooggetto}`: il sottooggetto ha una sottodidascalia con numero; il tutto compare anche nell’elenco relativo;
- `\subfloat [] [didascalia] {sottooggetto}`: il sottooggetto ha una sottodidascalia con numero, ma non compare nell’elenco relativo;
- `\subfloat [] [] {sottooggetto}`: il sottooggetto ha una sottodidascalia che consiste solo del numero, ma non compare nell’elenco relativo;
- `\subfloat [didascalia_breve] [didascalia] {sottooggetto}`: il sottooggetto ha la didascalia numerata e il contenuto di *didascalia\_breve* compare nell’elenco relativo con il numero;
- `\subfloat [didascalia_breve] [] {sottooggetto}`: il sottooggetto ha la didascalia che consiste solo del numero e il contenuto di *didascalia\_breve* compare nell’elenco relativo con il numero.

Ciascun *sottooggetto* può contenere un comando `\label` per potersi riferire in seguito a esso.

`\ContinuedFloat`

Se una figura o tabella è molto grande può essere spezzata in due o più ambienti dello stesso tipo successivi, dando questo comando all'inizio del secondo e di quelli che eventualmente seguono. Nella didascalia dei successivi è conveniente dare il comando `\caption` con argomento opzionale vuoto e ripetere il comando `\label` per potersi riferire ai numeri dei sottooggetti. Un esempio è nella Tabella 19.2 e il risultato si trova nella Figura 19.1. Il comando è, in realtà, del pacchetto `caption`.

---

**Tabella 19.2** Esempio di figura con quattro sottofigure spezzata in due parti.

---

```

\begin{figure}[htp]
\centering
\subfloat[] [] {...codice...}
\quad
\subfloat[] [] {...codice...}
\caption{Didascalia che descrive le prime due sottofigure.}
\label{fig:exa}
\end{figure}
\begin{figure}[htp]
\ContinuedFloat
\centering
\subfloat[] [] {...codice...}
\quad
\subfloat[] [] {...codice...}
\caption[] {Didascalia che descrive le altre due sottofigure.}
\label{fig:exa}
\end{figure}

```

---

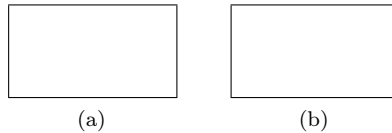


Figura 19.1: Didascalia che descrive le prime due sottofigure.

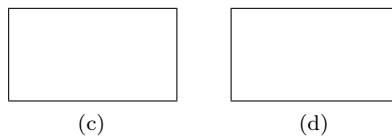


Figura 19.1: Didascalia che descrive le altre due sottofigure.



**paralist**

```
\usepackage[\langle opzione \rangle,...]{paralist}
```

Il pacchetto `paralist` è dedicato agli ambienti per comporre liste numerate o puntate. Fra le altre funzionalità ha anche quelle del pacchetto `enumerate`, che quindi non va caricato insieme a quello che descriveremo ora.

**20.1 — Opzioni**

Nel seguito, se le opzioni sono descritte in coppia la prima è quella scelta normalmente (*default*), cioè non è necessario specificarla.

```
newitem | olditem
```

Si può dare all'ambiente `itemize` come argomento opzionale la *marca*. Per esempio, con `\begin{itemize}[$\star$]` verrà usato il simbolo ‘ $\star$ ’ come marcatore degli elementi della lista. Con l'opzione `olditem` l'ambiente rimane invariato.

```
newenum | oldenum
```

Si può dare all'ambiente `enumerate` come argomento opzionale il modo di comporre il numero. La funzionalità è equivalente a quella del pacchetto `enumerate` e si rimanda al capitolo relativo per la descrizione. Con l'opzione `oldenum` l'ambiente rimane invariato.

```
alwaysadjust
```

Lo spazio riservato per le etichette negli ambienti `itemize`, `enumerate`, `compactitem` e `compactenum` è ampliato o ridotto a seconda dell'ampiezza dell'etichetta stessa.

```
neveradjust
```

L'opzione è ignorata se viene specificata quella precedente. Non modifica l'ampiezza per le etichette nemmeno per quelle specificate manualmente con gli argomenti opzionali.

```
neverdecrease
```

Se si usa l'opzione `alwaysadjust` impedisce che l'ampiezza delle etichette venga diminuita. Altrimenti comandi come `\begin{enumerate}` e `\begin{enumerate}[1.]` sarebbero composti in modo diverso.

```
defblank
```

Definisce gli ambienti `inparablank` e `asparablank`, si veda più avanti.

```
pointlessenum
```

Sopprime il punto finale nelle numerazioni dell'ambiente `enumerate` (senza argomenti opzionali). La numerazione negli ambienti annidati è uniforme: sarà quindi del tipo ‘1’, ‘1.1’, ‘1.1.1’.

```
pointedenum
```

Analoga alla precedente, ma la numerazione sarà del tipo ‘1.’, ‘1.1.’, ‘1.1.1.’.

---

```
flushright | flushleft
```

L'opzione normale (`flushright`) mantiene il comportamento usuale delle etichette negli ambienti per liste, cioè l'etichetta viene composta allineata a destra rispetto allo spazio assegnato. Con l'opzione `flushleft` l'allineamento è a sinistra.

```
cfg | nocfg
```

Normalmente `paralist` legge, se è presente, il file `paralist.cfg`. Con l'opzione `nocfg` la lettura non viene eseguita.

## 20.2 — Ambienti

Oltre alla possibile modifica già descritta degli ambienti `itemize` e `enumerate`, ci sono altri ambienti definiti da questo pacchetto.

```
\begin{asparaenum}[(testo)]
```

Produce una lista in cui gli elementi sono composti come se fossero normali paragrafi. L'argomento opzionale è lo stesso che si può dare all'ambiente `enumerate`. L'esempio si può vedere nella Tabella 20.1.

---

**Tabella 20.1** Esempi d'uso degli ambienti del pacchetto `paralist`

---

Questo testo appare prima  
della lista.

```
\begin{asparaenum}
\item Questo è il primo
elemento della lista,
abbastanza lungo per dover
andare a capo.
```

```
\item Questo è il secondo
elemento della lista. Andrà
a capo anche questo.
\end{asparaenum}
```

Questo testo appare dopo la  
lista.

Questo testo appare prima della lista.

1. Questo è il primo elemento della lista,  
abbastanza lungo per dover andare a capo.

2. Questo è il secondo elemento della lista.

Andrà a capo anche questo.

Questo testo appare dopo la lista.

---

Questa lista è costituita da  
elementi che rimangono in un  
paragrafo:

```
\begin{inparaenum}[(i)]
\item questo è il primo
elemento della lista,
\item questo è il secondo
elemento della lista.
\end{inparaenum}
```

Questo testo appare dopo  
la lista.

Questa lista è costituita da elementi che  
rimangono in un paragrafo: (i) questo è il  
primo elemento della lista, (ii) questo è il  
secondo elemento della lista.

Questo testo appare dopo la lista.

---

```
\begin{inparaenum}[(testo)]
```

L'ambiente produce una lista numerata, con lo stesso argomento opzionale possibile per l'ambiente `enumerate`, ma composta all'interno del paragrafo in cui si trova. Si noti che, a differenza di quanto si fa di solito, il comando `\item` non deve essere preceduto da una riga vuota.

```
\begin{compactenum}[\langle testo \rangle]
```

Produce una lista numerata senza le spaziature verticali aggiunte dall'usuale ambiente `enumerate`. L'argomento opzionale è lo stesso possibile per l'ambiente `enumerate`.

```
\begin{asparaitem}[\langle testo \rangle]
\begin{inparaitem}[\langle testo \rangle]
\begin{compactitem}[\langle testo \rangle]
```

I tre ambienti sono l'analogo dei precedenti, ma per `itemize` invece che per `enumerate`. L'argomento opzionale definisce il simbolo per la marcatura degli elementi della lista.

```
\begin{asparadesc}
\begin{inparadesc}
\begin{compactdesc}
```

I tre ambienti sono l'analogo dei precedenti rispetto all'ambiente `description`. Ricordare che in questi ambienti il comando `\item` *deve* avere l'argomento opzionale.

```
\begin{asparablank}
\begin{inparablank}
```

Sono ambienti richiesti all'autore del pacchetto da parte di utenti di LyX; chi legge questa guida non usa LyX e quindi non se ne serve. Sono disponibili solo dando l'opzione `defblank`.

### 20.3 — Comandi

I comandi che si descriveranno possono andare in un file `paralist.cfg` e in questo modo verranno eseguiti per ogni documento che carichi il pacchetto `paralist`.

```
\pointedenum | \pointlessenum
```

Sono analoghi alle opzioni che si possono dare al pacchetto, da usare localmente. Per esempio, in un documento in cui è in vigore l'opzione `pointedenum`, si può ottenere un ambiente in cui i numeri non siano seguiti dal punto nel modo seguente:

```
\newenvironment{myenum}
  {\pointlessenum\begin{enumerate}}
  {\end{enumerate}}
```

```
\paradescriptionlabel
```

Le etichette nell'ambiente `description` sono composte secondo la definizione del comando `\descriptionlabel`. Il comando in oggetto regola il formato delle etichette negli ambienti `asparadesc` e `inparadesc`. La sua definizione usuale è equivalente a

```
\newcommand{\paradescriptionlabel}[1]{\normalfont\bfseries#1}
```

e può essere modificata con `\renewcommand`.

```
\setdefaultitem{\langle testo_1 \rangle}{\langle testo_2 \rangle}{\langle testo_3 \rangle}{\langle testo_4 \rangle}
```

Possiamo, con questo comando, modificare le marche per i vari ambienti di tipo `itemize` annidati. La definizione delle classi standard è equivalente a

```
\setdefaultitem{\textbullet}{\normalfont}{\bfseries\textendash}
{\textasteriskcentered}{\textperiodcentered}
```

I quattro argomenti sono precisamente le marche da usare. Volendo modificarne solo una o più, è possibile lasciare gli altri argomenti vuoti:

```
\setdefaultitem{}{\$ \trianglerightright \$}{-}{-}
```

cambia solo le marche negli ambienti di tipo `itemize` di secondo livello.

```
\setdefaultenum{<testo1>}{<testo2>}{<testo3>}{<testo4>}
```

Possiamo, con questo comando, modificare le numerazioni per i vari ambienti di tipo `enumerate` annidati. La definizione delle classi standard è equivalente a

```
\setdefaultenum{1.}{(a)}{i.}{A.}
```

e come per il comando precedente è possibile lasciare un argomento vuoto per non modificare la numerazione nel livello corrispondente.

```
\setdefaultleftmargin{<dim1>}{<dim2>}{<dim3>}{<dim4>}{<dim5>}{<dim6>}
```

Possiamo, con questo comando, modificare il margine sinistro negli ambienti di composizione di liste. Ciascun argomento, che deve essere una dimensione, specifica l'aumento di margine in una lista annidata rispetto alla precedente. Le classi standard definirebbero

```
\setdefaultleftmargin{2.5em}{2.2em}{1.87em}{1.7em}{1em}{1em}
```

È possibile modificare solo uno o più aumenti di margine lasciando gli argomenti corrispondenti vuoti.

#### 20.4 — Parametri

Il pacchetto `paralist` mette a disposizione alcuni parametri dimensionali per modificare la composizione delle liste con gli ambienti `compactitem` e `compactenum`. Vanno modificati con `\setlength` o `\addtolength`.

```
\pltopsep
```

Spazio verticale tra il primo elemento e il testo precedente.

```
\plpartopsep
```

Spazio verticale aggiunto a `\pltopsep` nel caso la lista cominci un nuovo paragrafo.

```
\plitemsep
```

Spazio verticale fra gli elementi della lista.

```
\plparsep
```

Spazio verticale fra i paragrafi all'interno di un elemento della lista (è il `\parskip` locale all'ambiente).

**booktabs**

```
\usepackage{booktabs}
```

Il pacchetto non ha opzioni. Il suo scopo è di migliorare l'aspetto delle tabelle, che con i comandi usuali di L<sup>A</sup>T<sub>E</sub>X a volte lasciano a desiderare.

Chi impiega `booktabs` deve scordarsi di usare filetti verticali e limitare al massimo l'uso di filetti orizzontali. In effetti è buona pratica tipografica non usare *quasi mai* filetti verticali nelle tabelle.

**21.1 — Comandi per i filetti orizzontali**

Questi comandi vanno, eccezion fatta per `\toprule`, dopo il comando di fine riga `\`. Per i primi quattro, gli argomenti opzionali definiscono lo spessore della riga, ma non è il caso di usarli, lo spessore predefinito dall'autore va più che bene.

```
\toprule[⟨dimen⟩
\midrule[⟨dimen⟩
\bottomrule[⟨dimen⟩]
```

Creano filetti orizzontali con spaziature appropriate. Il primo va usato in testa, l'ultimo in coda; il secondo tipo di filetto serve a separare le parti di una tabella, non lo si usi indiscriminatamente dopo ogni riga.

```
\cmidrule[⟨dimen⟩]{⟨intervallo⟩}
```

È l'analogo di `\cline`, per creare un filetto che copra solo un certo intervallo di colonne. Si vedano gli esempi, anche per la discussione di una particolarità di questo comando.

```
\addlinespace[⟨dimen⟩]
```

Aggiunge una spaziatura fra le righe. Può anche essere usato prima o dopo uno dei comandi precedenti. Quindi non è proibito scrivere

```
...
⟨contenuto di una riga⟩\
\addlinespace
\midrule
\addlinespace
⟨contenuto di una riga⟩\
...
```

**21.2 — Esempi**

Consideriamo, come fa l'autore, uno degli esempi di tabella portati da Leslie Lamport nel suo manuale di L<sup>A</sup>T<sub>E</sub>X e, a fianco, mostriamo la versione con `booktabs`.

gnats	gram	\$13.65
	each	.01
gnu	stuffed	92.50
emu		33.33
armadillo	frozen	8.99

Item		Price (\$)
Animal	Description	
Gnats	per gram	13.65
	each	0.01
Gnu	stuffed	92.50
Emu	stuffed	33.33
Armadillo	frozen	8.99

Non c'è dubbio quale delle due sia migliore, sia dal punto di vista estetico che da quello specificamente informativo. Come giusto, nella tabella di sinistra c'è un'intestazione che spiega ciò che si troverà nelle colonne; le unità di misura, se necessario, vanno in questa intestazione. I filetti verticali servono solo a confondere le idee, quelli orizzontali non aggiungono nulla all'informazione.

Vediamo però come migliorare leggermente la resa della tabella.

Item		Price (\$)
Animal	Description	
Gnats	per gram	13.65
	each	0.01
Gnu	stuffed	92.50
Emu	stuffed	33.33
Armadillo	frozen	8.99

Osservando attentamente si noterà che il filetto che raggruppa due elementi dell'intestazione è leggermente più stretto; ciò si ottiene specificando subito dopo `\cmidrule` un argomento opzionale fra *parentesi tonde*: stiamo dicendo che il filetto va 'rasato' a sinistra e a destra. L'unica differenza fra questa realizzazione della tabella e la precedente è l'uso in questa di `\cmidrule(1r){1-2}` e nell'altra di `\cmidrule{1-2}`.

Ecco un altro esempio, in cui rasiamo l'inizio e la fine di quasi tutti i filetti.

	test 1			test 2		
	Liftoff	Liftoff	Liftoff	Liftoff	Liftoff	Liftoff
	Angle	Declination	Speed	Angle	Declination	Speed
$\phi$	$\theta$ (°)	Angle (°)	(m/s)	$\theta$ (°)	Angle (°)	(m/s)
0.3	84.6	43.8	0.716	84.6	43.8	0.716
0.6	72.5	48.7	0.842	72.5	48.7	0.842

```

\begin{tabular}{@{}*{7}{c}@{}}
\toprule
& \multicolumn{3}{c}{test 1} & \multicolumn{3}{c}{test 2} \\
\cmidrule(1r){2-4} \cmidrule(1){5-7}
& Liftoff & Liftoff & Liftoff & Liftoff & Liftoff & Liftoff \\
& Angle & Declination & Speed & Angle & Declination & Speed \\
\phi & \theta (°) & Angle (°) & (m/s) & \theta (°) & Angle (°) & (m/s) \\
\cmidrule(r){1-1} \cmidrule(1r){2-4} \cmidrule(1){5-7}
0.3 & 84.6 & 43.8 & 0.716 & 84.6 & 43.8 & 0.716 \\
0.6 & 72.5 & 48.7 & 0.842 & 72.5 & 48.7 & 0.842 \\
\bottomrule
\end{tabular}

```

Si noterà come ai comandi `\cmidrule` sia specificata una rasatura solo da una sola parte, se sono coinvolte la prima o l'ultima colonna che finiscono a filo del margine della tabella. Le rasature sono necessarie per specificare il raggruppamento delle colonne.

### 21.3 — Linee guida per comporre una tabella

Traduco liberamente le indicazioni di Simon Fear, autore del pacchetto `booktabs`.

1. Non usate *mai* filetti verticali.
2. Non usate *mai* doppi filetti.
3. Indicate le unità di misura nell'intestazione della colonna.

4. Fate sempre precedere il punto decimale da una cifra; usando la virgola decimale l'uso è sempre stato questo, molti al giorno d'oggi impiegano anche in Italia il punto decimale. Si sia coerenti all'interno dello stesso documento e si osservi sempre la regola appena scritta.
5. Non usate mai virgolette o qualsiasi altra convenzione per ripetere un valore della riga precedente. In molti casi basta lasciare uno spazio bianco; se uno spazio bianco non va, allora ripetete il valore.

Aggiungo un'altra regola: non siate avari nelle intestazioni, magari rimandando le spiegazioni nella didascalia, meglio una riga in più che una possibile ambiguità. Non si deve però eccedere con lunghi testi: la spiegazione del significato di un simbolo, per esempio, va certamente nella didascalia.

Nelle pagine precedenti avete certamente notato esempi di filetti verticali; erano solo per mostrare esempi. Quasi tutte quelle tabelle avrebbero potuto essere composte senza filetti verticali; la regola 1 va interpretata *cum grano salis*: se l'informazione va letta per righe, non ci devono essere filetti verticali. A volte una tabella lunga e stretta può utilmente essere divisa a metà e, in tal caso, un filetto per separare le due metà è certamente indicato. Non è però più possibile usare le funzionalità di `booktabs`.

<b>Valori di seno e coseno</b>					
	0	$\pi/2$	$\pi$	$3\pi/2$	$2\pi$
seno	0	1	0	-1	0
coseno	1	0	-1	0	1

```

\begin{tabular}{l*{5}{>{<}c<{<}}
\toprule
\multicolumn{6}{c}{%
\textbf{Valori di seno e coseno}}\}
\midrule
& 0 & \pi/2 & \pi & 3\pi/2 & 2\pi \\
\cmidrule{2-6}
seno & 0 & 1 & 0 & -1 & 0 \\
coseno & 1 & 0 & -1 & 0 & 1 \\
\bottomrule
\end{tabular}

```

#### 21.4 — Compatibilità

Il pacchetto `booktabs` è compatibile sia con `colortbl` che con `longtable`. Inoltre, caricando `array`, sono disponibili le estensioni fornite da quel pacchetto.

## ifpdf

`\usepackage{ifpdf}`

Il pacchetto mette a disposizione il condizionale `\ifpdf` che risulta vero se il compilatore è `pdflatex`, falso se il compilatore è `latex`. Le recenti distribuzioni di  $\TeX$  usano in realtà, in entrambi i casi, il programma `pdftex`, lasciando al sistema operativo di stabilire quali parametri passargli nel caso sia chiamato con uno dei due nomi precedenti. Per esempio, chiamando `pdflatex`, il parametro interno `\pdfoutput` verrà inizializzato a 1.

Fino a qualche tempo fa era comune definirsi da sé il condizionale con costruzioni come

```
\newif\ifpdf
\ifx\pdfoutput\undefined
  \pdffalse
\else
  \pdftrue
\fi
```

Questo modo di procedere *non* funziona con le più recenti distribuzioni, proprio perché la sequenza di controllo `\pdfoutput` è *sempre* definita. Va invece usato questo pacchetto che definisce il condizionale `\ifpdf` nel modo corretto.

### 22.1 — Esempi

Supponiamo di voler passare a `hyperref` opzioni diverse nel caso compiliamo con `latex` o con `pdflatex`; scriveremo cose come

```
\ifpdf
  \usepackage{thumbpdf}
  \usepackage[plainpages=false,pdfpagelabels]{hyperref}
  \hypersetup{colorlinks,breaklinks}
\else
  \usepackage{hyperref}
  \hypersetup{colorlinks,backref,hyperindex,breaklinks}
\fi
```

Si ricorda che non va *mai* passata ai pacchetti `hyperref`, `graphicx` e `color` un'opzione esplicita come `pdftex` o `dvips`, perché i pacchetti sanno riconoscere automaticamente quale delle due caricare.

Se fosse necessario il condizionale per passare opzioni diverse a `\documentclass`, il pacchetto `ifpdf` dovrà essere caricato prima e, in tal caso, si dovrà usare `\RequirePackage`:

```
\RequirePackage{ifpdf}
\ifpdf
  \documentclass[a4paper]{article}
\else
  \documentclass[textures,a4paper]{article}
\fi
```

perché prima di `\documentclass` il comando `\usepackage` non è ancora definito. L'esempio non è molto realistico, ma al giorno d'oggi è difficile pensare ad altri casi in cui questo sia necessario. In questo caso l'utente vuole passare ai pacchetti l'opzione globale `textures` nel caso si compili con `latex`; non vuole farlo, ovviamente, nel caso si compili con `pdflatex`.



Il condizionale `\ifpdf` può essere usato anche in altre situazioni all'interno del documento, probabilmente nascosto dentro un comando personale. Si può naturalmente usare una costruzione come `\boolean{pdf}` se si è caricato `ifthen`. Per esempio, dando

```
\newcommand*{\ack}{Compilato il giorno \today{}} con
\texttt{\ifpdf pdf\fi latex}}
```

il comando `\ack` stamperà qualcosa come

```
Compilato il giorno 28 gennaio 2011 con pdflatex
```

### 22.2 — Pacchetti analoghi

Esistono anche i pacchetti `ifxetex`, `ifvtex` e `ifluatex` che mettono a disposizione i condizionali `\ifxetex`, `\ifvtex` e `\ifluatex` che risultano veri se si stanno usando le corrispondenti estensioni di  $\TeX$ .

## enumitem

`\usepackage[<opzione>,...] {enumitem} [2007/06/30]`

Il pacchetto `enumitem` è nato con scopi analoghi a `enumerate` e `paralist`; ne incorpora le funzionalità, quindi va probabilmente preferito a quei pacchetti, pur se la sintassi è diversa. La funzione essenziale è di modificare gli ambienti `enumerate`, `itemize` e `description`, quindi `paralist` rimane utile per gli ambienti all'interno dei capoversi. La versione che descriviamo qui è la 2, piuttosto diversa dalla precedente, per questo indichiamo la data di rilascio.

### 23.1 — Opzioni

`ignoreddisplayed`

L'opzione non modifica l'ambiente `trivlist` originale; non se ne consiglia l'uso.

`loadonly`

I tre ambienti fondamentali non sono modificati, ma se ne possono definire di nuovi con il comando `\newlist`.

`shortlabels`

È un'opzione che tenta una specie di compatibilità con il pacchetto `enumerate`. Non vale la pena di usarla: se si decide per `enumitem` meglio andare fino in fondo.

### 23.2 — Comandi

```
\setlist[<livello>]{<formato>}
\setenumerate[<livello>]{<formato>}
\setitemize[<livello>]{<formato>}
\setdescription{<formato>}
```

Questi comandi servono a impostare i parametri per gli ambienti `enumerate`, `itemize` e `description`. Qui *<livello>* è un numero fra 1 e 4, se non è espresso l'impostazione vale a tutti i livelli di annidamento. Le impostazioni date come argomento di `\setlist` valgono per i tre ambienti; invece `\setenumerate` vale solo per gli ambienti `enumerate` e similmente per `\setitemize` e `\setdescription`.

Il *<formato>* è un elenco di opzioni nella forma `chiave=valore` separate da virgole e i comandi possono essere ripetuti con effetto cumulativo. Vedremo nella sezione [23.3](#) le possibili chiavi e i valori che possono assumere.

```
\newlist{<nome>}{<tipo>}{<livello>}
\setlist[<nome>]{<formato>}
\setlist[<nome>,<livello>]{<formato>}
```

È possibile 'clonare' una delle tre liste con il comando `\newlist` che prende come primo argomento il nome del nuovo ambiente basato sull'ambiente *<tipo>* dato come secondo argomento; il terzo argomento è il massimo livello di annidamento del nuovo ambiente. Per esempio

```
\newlist{myenum}{enumerate}{2}
```

definisce un nuovo ambiente basato su `enumerate`, che ammette due livelli di annidamento; ovviamente useremo `\setlist` per definirlo nei dettagli in uno dei modi seguenti:

```
\setlist[myenum]{\formato}
\setlist[myenum,1]{\formato}
\setlist[myenum,2]{\formato}
```

Con il primo comando applicheremo  $\langle formato \rangle$  all'ambiente in entrambi i livelli di annidamento; con il secondo o il terzo,  $\langle formato \rangle$  sarà applicato solo al livello indicato. Il comando  $\setlist$  è *obbligatorio*, almeno per stabilire l'etichetta da assegnare agli elementi della lista che si sta definendo.

Non va usato  $\setlist$  con l'argomento opzionale per uno dei tre ambienti di base (*enumerate*, *itemize* e *description*), per i quali ci sono i comandi citati in precedenza.

```
\AddEnumerateCounter{\comando_1}{\comando_2}{\testo}
```

Con questo comando si 'registra' una rappresentazione di registri numerici che non sia di quelle usuali come  $\arabic$ ,  $\Alph$  e simili. Il  $\langle comando_1 \rangle$  e il  $\langle comando_2 \rangle$  sono rispettivamente il comando a livello utente e quello interno; per esempio il pacchetto definisce

```
\AddEnumerateCounter{\arabic}{\@arabic}{0}
\AddEnumerateCounter{\alph}{\@alph}{m}
\AddEnumerateCounter{\Alph}{\@Alph}{M}
\AddEnumerateCounter{\roman}{\@roman}{viii}
\AddEnumerateCounter{\Roman}{\@Roman}{VIII}
```

Ovviamente comandi simili andranno circondati da  $\makeatletter$  e  $\makeatother$ . Per esempio, il pacchetto *itnumpar* definisce una rappresentazione dei numeri con parole in italiano; se volessimo usare questa rappresentazione in una delle liste, per esempio con gli ordinali maschili, dovremmo dire

```
\AddEnumerateCounter{\ordinalem}{\@ordinalem}{quattordicesimo}
```

L'ultimo argomento indica quale sia una realizzazione 'massima' della rappresentazione. Dovremo quindi stimare quale fra le rappresentazioni dei numeri sia la più ampia in larghezza.

Per essere più rigorosi, i due comandi devono essere comandi con un argomento, il secondo deve prendere come argomento il valore di un contatore. Perciò, per registrare la rappresentazione testuale di un contatore senza caricare pacchetti esterni, sapendo che ci si può limitare a pochi valori, si può usare (non sono necessari  $\makeatletter$  e  $\makeatother$ )

```
\AddEnumerateCounter{\ord}{\innord}{Secondo}
\newcommand{\ord}[1]{\expandafter\innord\csname c@#1\endcsname}
\newcommand{\innord}[1]{\ifcase#1\or Primo\or Secondo\or
Terzo\or Quarto\or Quinto\or Settimo\or Ottavo\fi}
```

Vedremo nella sezione successiva come usare queste nuove rappresentazioni.

### 23.3 — Chiavi e valori

```
topsep= $\langle dimen \rangle$ 
partopsep= $\langle dimen \rangle$ 
parsep= $\langle dimen \rangle$ 
itemsep= $\langle dimen \rangle$ 
```

Con queste chiavi si controllano le spaziature verticali, i nomi corrispondono ai parametri usuali di  $\LaTeX$ : (1)  $\topsep$  è la spaziatura verticale per la separazione della lista dal contesto, viene usata sopra e sotto; (2)  $\partopsep$  è la spaziatura aggiuntiva nel caso la lista sia preceduta (o seguita) da una riga vuota nel documento  $\LaTeX$ ; (3)  $\parsep$  è la spaziatura verticale aggiunta quando c'è un cambio di capoverso in un elemento della lista; (4)  $\itemsep$  è la spaziatura verticale tra elementi della lista.

I valori dei quattro parametri in un documento della classe `article` a corpo 10 sono:

```
\topsep      8.0pt plus 2.0pt minus 4.0pt
\partopsep   2.0pt plus 1.0pt minus 1.0pt
\parsep      4.0pt plus 2.0pt minus 1.0pt
\itemsep     4.0pt plus 2.0pt minus 1.0pt
```

Li si modifichi con una certa cautela: azzerare `\topsep`, per esempio, impedisce di separare la lista dal contesto.

```
noitemsep
nolistsep
```

A queste chiavi non va dato un valore: con la prima si azzerano i valori di `\itemsep` e `\parsep`, mentre con la seconda si azzerano in blocco i quattro parametri di spaziatura visti in precedenza. È bene non usare il secondo per le liste normali, può essere utile per liste in contesti particolari.

```
leftmargin=<dimen>
rightmargin=<dimen>
listparindent=<dimen>
labelwidth=<dimen>
labelsep=<dimen>
itemindent=<dimen>
```

Queste chiavi servono a impostare le spaziature orizzontali. Con `leftmargin` si imposta la distanza del margine sinistro degli elementi della lista dal margine sinistro dell'ambiente circostante. Si ricordi che L<sup>A</sup>T<sub>E</sub>X definisce i sei parametri `\leftmargini`, `\leftmarginii`, `\leftmarginiii`, `\leftmarginiv`, `\leftmarginv` e `\leftmarginvi` e la classe ne imposta i valori; è poi compito dell'ambiente di lista dare, in base al livello di annidamento, il corretto valore a `\leftmargin`.

Non ci sono parametri, invece, per impostare il valore `\rightmargin`, che le classi usualmente mantengono nullo.

La chiave `listparindent` imposta il valore del rientro per i capoversi successivi al primo in un elemento della lista; questo rientro è, usualmente, nullo. Per esempio, si può ottenere un rientro uguale a quello dei capoversi normali con

```
listparindent=\parindent
```

fra le chiavi impostate nel *<formato>*.

Per capire gli ultimi tre, si deve ricordare che il parametro `\itemindent` contiene la distanza orizzontale del primo carattere di un elemento della lista dal margine sinistro (della lista stessa, ovviamente). A sinistra di questo primo carattere sarà posizionata l'etichetta, secondo una regola che al principio può lasciare perplessi.

La parte dedicata all'etichetta si compone di tre parti: da sinistra a destra, la prima è uno spazio vuoto di ampiezza `\labelindent`; la seconda è una 'box' di ampiezza `\labelwidth`; la terza uno spazio vuoto di ampiezza `\labelsep - \itemindent`. L'etichetta stessa va nella 'box', allineata usualmente rispetto al margine destro di essa. Se la somma di queste tre dimensioni è maggiore del valore di `\leftmargin`, questa costruzione sposterà a sinistra. Si noti che il terzo spazio vuoto potrebbe risultare negativo e, in questo caso, l'etichetta riempirà parte del rientro del primo carattere dell'elemento della lista.

```
font=<font>
```

Con questa chiave si sceglie in che tipo di carattere deve essere composta l'etichetta. Può essere, per esempio, un comando come `\itshape`; meglio ancora, può ricevere come valore `\upshape` in modo che l'etichetta sia composta in carattere dritto indipendentemente dal contesto. La specificazione potrebbe anche contenere comandi di cambio di colore, se si è caricato il pacchetto `color` o `xcolor`.

`label=<testo>`

Si imposta l’etichetta; ha senso naturalmente per gli ambienti basati su `enumerate` e `itemize`. Nel caso del primo tipo, nel `<testo>` dovrà comparire la rappresentazione di un contatore; non occorre sapere quale, perché con un comando del tipo `\arabic*` si impone di usare il contatore relativo al livello di annidamento attuale, cioè `enumi`, `enumii`, `enumiii` oppure `enumiv`.

`label*=<testo>`

I comandi contenuti in `<testo>` vengono aggiunti all’etichetta del livello superiore. L’esempio classico è quello di liste numerate di tipo “legale”, che vanno numerate con “1”, “1.1”, “1.1.1” e così via. Un modo semplice per ottenerle è di definire

```
\newlist{legal}{enumerate}{4}
\setlist[legal]{label*=\arabic*}
\setlist[legal,1]{label=\arabic*}
```

Come si vede, si definisce uno schema di etichetta valido per tutti i livelli, modificando poi quella per il primo livello.

`ref=<testo>`

Se non si specifica la chiave `ref`, il valore del riferimento all’etichetta coincide con quello dato dalla chiave `label` (o quello usuale, se non si specifica nemmeno questa chiave). Usando `ref` si può stabilire un formato diverso: con

```
\begin{enumerate}[label=\emph{\alph*},ref=\emph{\alph*}]
```

l’etichetta del primo elemento sarebbe “a”, ma un riferimento (tramite i comandi `\label` e `\ref`) darebbe “a”. La sintassi è identica a quella per `label`, ma non c’è la versione con l’asterisco.

`leftmargin=*`

Con questo valore speciale, il valore di `\leftmargin` viene definito come l’ampiezza dell’etichetta sommato a `\labelsep`. Nel caso di un ambiente basato su `enumerate`, l’ampiezza dell’etichetta è quella di ‘0’ se il numero è rappresentato con `\arabic`, di ‘m’ se il numero è rappresentato con `\alph` e di ‘viii’ se il numero è rappresentato con `\roman`; nel caso di `\Alph` e `\Roman` l’ampiezza è quella di ‘M’ e ‘VIII’, rispettivamente.

`labelsep=*`

Analogo al caso precedente; qui il valore di `\labelsep` viene definito come la differenza fra `\leftmargin` e `\labelwidth`.

`widest=<testo>`

Se si usa `leftmargin=*` o `labelsep=*`, può essere utile specificare l’ampiezza massima prevista della parte numerica dell’etichetta. Per esempio, se la lista è un `enumerate` a numeri arabi con dodici elementi, l’impostazione corretta sarà, per esempio,

```
\begin{enumerate}[leftmargin=*,label=\arabic*.,widest=00]
```

In questo modo, il margine sinistro dei numeri con due cifre sarà allineato al margine sinistro dell’ambiente circostante.

`labelindent=<dimen>`

Si può dare una dimensione per specificare, con `leftmargin=*` o `labelsep=*`, un rientro aggiuntivo dell’etichetta. È utile anche con ambienti basati su `description`.

```
start=<numero>
```

Ha senso solo per un ambiente basato su `enumerate`; specifica il numero da cui cominciare nel conteggio degli elementi.

```
resume
```

Riprende il conteggio da dove si è interrotto il precedente ambiente *con lo stesso nome*.

```
resume*
```

È analoga alla chiave senza asterisco, ma in questo caso vengono riprese tutte le impostazioni date all'ambiente precedente e non se ne possono dare altre. Non si abusi di queste due chiavi, è un caso molto raro che si debba riprendere un conteggio, ancora più raro che questo avvenga con un altro ambiente di tipo `enumerate` in mezzo.

```
beginpenalty=<numero>
midpenalty=<numero>
endpenalty=<numero>
```

Vengono inserite le penalità specificate rispettivamente prima della lista, fra gli elementi e alla fine della lista. Si faccia attenzione che questi valori vengono mantenuti anche per eventuali liste di livello superiore.

```
before=<testo>
before*=<testo>
after=<testo>
after*=<testo>
```

Il `<testo>` è codice che viene eseguito rispettivamente prima e dopo l'inizio della lista; per esempio, si potrebbe usare

```
before=\itshape
```

per comporre tutta la lista in corsivo. Si faccia però attenzione che questo influenzerebbe anche l'etichetta. Le chiavi con l'asterisco aggiungono il `<testo>` a eventuale codice ereditato da un'impostazione precedente; quelle senza asterisco lo sostituiscono. Per esempio, con

```
\newlist{alphen}{enumerate}{4}
\setlist[alphen]{label=\alph*}
\setlist[alphen,2]{before=\itshape}

\begin{alphen}
\item a
\item \begin{myen} [before=\small]
\item x
\end{alphen}
\item b
\item \begin{myen} [before*=\small]
\item y
\end{alphen}
\end{alphen}
```

Nel primo ambiente annidato il testo sarà dritto e a corpo ridotto, perché la specificazione `\itshape` non viene considerata; nel secondo ambiente annidato, il testo sarà corsivo e a corpo ridotto.

Si noti che questa aggiunta o sostituzione si intende al codice specificato nelle impostazioni dell'ambiente relative al livello di annidamento in cui ci si trova.

`align=right` | `left`

Con questa chiave si decide la posizione dell’etichetta rispetto alla ‘box’ di ampiezza `\labelwidth` che la contiene; se non la si specifica, la posizione è con allineamento al margine destro, con `align=left` si ottiene il contrario.

`style=sameline` | `nextline` | `multiline` | `unboxed`

Questa chiave è disponibile solo per gli ambienti basati su `description`. Nell’ambiente `description` senza specificazioni di questo tipo l’etichetta (definita tramite l’argomento opzionale a `\item`) viene inserita allineata al margine sinistro dell’ambiente circostante e separata di `\labelsep` dal testo che quindi, nel caso l’etichetta sia corta, sporge a sinistra rispetto alle righe successive. Specificando invece un valore per la chiave `style`, il testo della prima riga dell’elemento non sporge mai a sinistra rispetto alle righe successive. I quattro stili differiscono per il comportamento rispetto a etichette più larghe di `\leftmargin`: (1) con `sameline` il testo comincia sulla stessa riga dell’etichetta, più a destra; (2) con `nextline` il testo comincia sulla riga successiva; (3) con `multiline` l’etichetta viene composta in una `\parbox` di ampiezza `\leftmargin` (si eviti con cura questa bruttura); (4) con `unboxed` si ha un comportamento analogo a `sameline`, con la differenza che l’etichetta non è composta in una ‘box’ e quindi, se lunga più di una riga, può andare a capo (si eviti con cura anche questa bruttura).

`fullwidth`

Questa chiave non prende alcun valore. Il valore di `\leftmargin` è nullo e l’etichetta diventa semplicemente parte del testo; in pratica si scrivono capoversi ‘numerati’.

### 23.4 — Esempi

Va subito detto chiaramente che è in generale sconsigliabile annidare più di una lista dentro un’altra, se non è richiesto da consuetudini come quelle delle liste di tipo legale menzionate prima. Tuttavia l’impiego di liste numerate in modo particolare può essere indicato se queste liste hanno significati diversi.

Un caso è quello di liste da usare in enunciati di teoremi, dove è opportuno distinguere se si dà una lista di condizioni equivalenti, una di ipotesi oppure una di conclusioni:

```
\newlist{roster}{enumerate}{1}
\setlist[roster]{label=\upshape(\arabic*),noitemsep}
\newlist{rostera}{enumerate}{1}
\setlist[rostera]{label=\upshape(\alph*),noitemsep}
\newlist{rosteri}{enumerate}{1}
\setlist[rosteri]{label=\upshape(\roman*),noitemsep}
```

dove usiamo anche l’opzione `noitemsep` perché all’interno di un enunciato è bene evitare una spaziatura eccessiva fra elementi della lista. Non è possibile, con `enumitem`, definire questi ambienti in termini di uno solo, perché una volta dato l’argomento opzionale a `enumerate` non se ne possono passare altri; perciò, definendo per esempio

```
\newenvironment{roster}[1]
{\begin{enumerate}[label=\upshape(#1*)]}
{\end{enumerate}}
```

sarebbe possibile ottenere le variazioni con

```
\begin{roster}{\arabic}
\begin{roster}{\alph}
\begin{roster}{\roman}
```

ma non si potrebbe specificare un’eventuale chiave `resume`.

Un esempio più complicato: vogliamo sostituire i numeri romani “minuscoli” con numeri romani maiuscoli in carattere ridotto. Per prima cosa definiamo una nuova rappresentazione:

```

\makeatletter
\def\smallRoman#1{\expandafter\@smallRoman\csname c@#1\endcsname}
\def\@smallRoman#1{\textnormal{\scshape\romannumeral #1}}
\AddEnumerateCounter{\smallRoman}{\@smallRoman}
  {\textnormal{\scshape VIII}}
\makeatother

```

A questo punto possiamo usare

```

\begin{enumerate}[label=\smallRoman*.]
\item primo elemento
\item secondo elemento
\end{enumerate}

```

Si faccia attenzione che questo metodo può fallire se il carattere tipografico in uso non ha il maiuscolo. Si potrebbe anche agire più semplicemente scrivendo

```

\begin{enumerate}[label=\textnormal{\scshape\roman*}]

```

ma questo contravverrebbe al principio base di L<sup>A</sup>T<sub>E</sub>X: separare la forma dal contenuto.

Le classi standard usano numeri romani minuscoli per le liste annidate al terzo livello; volendo modificare il comportamento, adoperando numeri romani maiuscoli in corpo ridotto, basta scrivere

```

\setenumerate[3]{label=\smallRoman*.}

```

dopo aver registrato la rappresentazione con `\AddEnumerateCounter`. Questo non significa che sia opportuno usare liste annidate fino al terzo livello.

Un esempio già discusso con `enumerate`:

```

\newlist{axiomslist}{enumerate}{1}
\setlist[axiomslist]{label=\upshape(\axiomtag$_{\arabic*}$),
  leftmargin=*,widest=\axiommax,align=left}
\newenvironment{axioms}[2][9]
  {\def\axiomtag{#2}\def\axiommax{#1}\begin{axiomslist}}
  {\end{axiomslist}}

```

L'ambiente `axioms` prende come argomento opzionale un numero (default 9), nel caso gli elementi della lista siano più di 9, e come argomento opzionale la parte comune dell'etichetta. Così le chiamate

```

\begin{axioms}[99]{SV}
\begin{axioms}{M}

```

producono liste del tipo

<pre> ... (SV<sub>12</sub>) <math>(u + v) + w = u + (v + w)</math>, (SV<sub>13</sub>) <math>u + 0 = u, 0 + u = u</math>, ... (M<sub>1</sub>) primo assioma (M<sub>2</sub>) secondo assioma </pre>
---

Per altri esempi si veda l'estesa documentazione del pacchetto. Il pacchetto è compatibile con `paralist`, ma va caricato dopo questo e, naturalmente, si potranno usare solo le funzioni di `paralist` per le liste “in corpo” (sezione 20.2).



**url**

```
\usepackage[\langle opzione \rangle,...] {url}
```

Il pacchetto `url` serve per scrivere senza fatica i cosiddetti URL (*uniform resource locator*) o, più precisamente, URI (*uniform resource identifier*), così comuni al giorno d'oggi. Il pacchetto viene automaticamente caricato da `hyperref`.

**24.1 — Opzioni**

```
obeyspaces
```

Il comportamento normale di `url` è di ignorare gli spazi nell'argomento dei comandi che definisce; con questa opzione, invece, gli spazi non sono ignorati; tuttavia potrebbero comparire spazi non desiderati dopo una barra rovescia `'\`.

```
hyphens
```

Le parti di testo controllate da `url` non vengono mai spezzate dopo un trattino; con questa opzione sì. Sconsigliabile.

```
spaces
```

Va data insieme all'opzione `obeyspaces`, per permettere di andare a capo dopo uno spazio. Sconsigliabile.

```
lowtilde
```

Normalmente la tilde appare come `~`. Con questa opzione appare come `˜`.

**24.2 — Comandi**

```
\url{\langle uri \rangle}
\url{\langle char \rangle\langle uri \rangle\langle char \rangle}
```

Il comando `\url` ha due forme; la prima è la più comune e va usata se l'argomento `\langle uri \rangle`, cioè l'URI in questione, non contiene graffe non bilanciate. Nella seconda forma `\langle char \rangle` è un carattere che non compare in `\langle uri \rangle` e non è una graffa aperta né uno spazio. In tal caso possono comparire nell'URI anche graffe non bilanciate.

In entrambe le forme, se l'URI contiene `%` o `^^` oppure termina con `\`, non può essere usato come argomento di un altro comando.

```
\urldef{\langle comando_1 \rangle}{\langle comando_2 \rangle}{\langle uri \rangle}
```

Se si ha bisogno assoluto di adoperare un URI come argomento a un comando e `\url` non funziona, lo si può 'bloccare' definendo un comando apposito (robusto):

```
\newcommand{\mioURI}{}
\urldef\mioURI\url{http://a.b.cd/abc%20def}
```

e usare al posto dell'URI problematico il comando `\mioURI`. Si faccia *molta* attenzione, perché il pacchetto non esegue alcun controllo sull'esistenza del comando che diamo come primo argomento. Perciò è buona norma precedere l'uso di `\urldef` con `\newcommand`. Il secondo comando è, usualmente, `\url`; si veda più avanti.

```
\urlstyle{\langle tt | rm | sf | same \rangle}
```

Con questo comando si stabilisce come vengono composti gli URI; lo stile `same` usa il font corrente, gli altri dovrebbero spiegarsi da sé; se non si dà il comando, lo stile è `tt`.

```
\DeclareUrlCommand<comando>{(testo)}
```

È possibile dichiarare nuovi comandi simili a `\url` con le stesse due forme. Per esempio, il pacchetto usa

```
\DeclareUrlCommand\url{}
\DeclareUrlCommand\path{\urlstyle{tt}}
```

in modo che dopo `\urlstyle{rm}` l'argomento di `\url` sarebbe composto in tondo, ma quello di `\path` continuerebbe a essere composto in carattere dattilografico. Alcuni stili chiedono che gli indirizzi di posta elettronica siano composti nel font corrente, basterebbe allora definire

```
\DeclareUrlCommand\emailURI{\urlstyle{same}}
```

Nel secondo argomento di questo comando possono andare altre cose, che vengono eseguite prima della composizione dell'URI; per esempio si potrebbe definire

```
\DeclareUrlCommand\emailuri{\texttt{mailto:}\urlstyle{tt}}
```

per poi scrivere `\emailuri{pippo@topolinia.tp}` ottenendo direttamente l'URI nella forma `mailto:pippo@topolinia.tp`. Meglio non approfittarne troppo.

Tutti i comandi definiti con `\DeclareUrlCommand` possono poi essere usati come secondo argomento di `\urldef`.

### 24.3 — Interazione con `hyperref`

Un'altra utile guida su  $\text{\LaTeX}$ , adatta a lettori più esperti nel linguaggio, può essere trovata all'URI

```
http://profs.sci.univr.it/~gregorio/introtex.pdf
```

Dal momento che il documento usa `hyperref`, questo URI che è stato inserito tramite

```
\url{http://profs.sci.univr.it/~gregorio/introtex.pdf}
```

diventa automaticamente un collegamento ipertestuale. Per evitarlo basta definire un nuovo comando con, per esempio,

```
\DeclareUrlCommand\uri{}
```

e l'uso di `\uri` non creerà collegamenti.

## tocloft

`\usepackage[titles | subfigure]{tocloft}`

Le classi standard di L<sup>A</sup>T<sub>E</sub>X non permettono di modificare facilmente i parametri di impaginazione dell'indice generale e degli elenchi di figure e tabelle. Un caso che di frequente causa problemi è quello del numero eccessivo di sezioni in un capitolo (più di nove), perché i numeri delle sezioni si avvicinano troppo ai titoli. Il pacchetto `tocloft` permette di aggiustare senza troppa fatica i parametri rilevanti e di agire in modo molto libero sulla composizione degli indici; fra questi *non* è compreso l'indice analitico che è una cosa molto diversa.

In linea di principio queste modifiche possono essere portate anche senza pacchetti aggiuntivi: il formato tipico di una linea del documento `.toc`, per esempio, è

```
\contentsline {chapter}{\numberline {1}babel}{7}
\contentsline {chapter}{\numberline {1}babel}{7}{chapter.1}
```

(riporto sia la forma senza `hyperref` sia quella modificata che permette i collegamenti ipertestuali). Il trucco è che `\contentsline{chapter}` richiama il comando `\l@chapter`, così come la combinazione `\contentsline{section}` richiama il comando `\l@section` e così via. A loro volta i comandi `\l@chapter` e simili fanno uso di funzioni e parametri appositi, come `\p@numwidth` che contiene la dimensione riservata ai numeri.

Con `tocloft` si può accedere a questi parametri e funzioni in modo meno complicato. Si tenga però presente che questo pacchetto non è compatibile con le classi AMS; la classe `memoir` ha invece sistemi simili già previsti.

### 25.1 — Convenzioni

Poiché `tocloft` introduce molti comandi dal nome simile, faremo le seguenti convenzioni:

- *<ext>* significa `toc`, `lof` oppure `lot`; quindi, per esempio, `\cft<ext>titlefont` sta per uno dei comandi `\cfttoctitlefont`, `\cftloftitlefont` o `\cftlottitlefont`;
- *<cnt>* sta per il nome abbreviato di un contatore relativo alle divisioni del documento, secondo la tabella 25.1; quindi, per esempio, `\cft<cnt>indent` può essere `\cftpartindent`, `\cftchapindent`, `\cftfigindent`, eccetera.

Si noti però che con `\newlistof` si definiscono altre stringhe che possono andare al posto di *<ext>* e *<cnt>*.

I comandi di `tocloft` sono molti: alcuni sono parametri di lunghezza, altri dichiarazioni e altri ancora semplici contenitori. I parametri di lunghezza si modificano con `\setlength` o `\addtolength`; nel seguito menzioneremo solo il primo modo, è sottinteso che ogni altro metodo per impostare un parametro dimensionale va bene. I 'contenitori' vanno modificati con `\renewcommand`, cercheremo di specificarlo per ogni comando descritto.

**Tabella 25.1** Abbreviazioni dei nomi di contatori; si ricordi che il contatore `chapter` è collegato al comando `\chapter` e similmente per gli altri.

Abbreviazione	Contatore	Abbreviazione	Contatore
<code>part</code>	<code>part</code>	<code>para</code>	<code>paragraph</code>
<code>chap</code>	<code>chapter</code>	<code>subpara</code>	<code>subparagraph</code>
<code>sec</code>	<code>section</code>	<code>fig</code>	<code>figure</code>
<code>subsec</code>	<code>subsection</code>	<code>tab</code>	<code>table</code>
<code>subsubsec</code>	<code>subsubsection</code>		

## 25.2 — Opzioni

`titles`

Con questa opzione al pacchetto si lascia la composizione dei titoli a L<sup>A</sup>T<sub>E</sub>X, senza poter intervenire con i comandi appositamente definiti da `tocloft`. Va data se si usa un pacchetto come `titlesec` che agisce già anche sui titoli degli indici e potrebbe essere confuso dal sistema introdotto da `tocloft`.<sup>1</sup>

`subfigure`

Secondo la documentazione, va data se e solo se si usa il pacchetto `subfigure`. Dovrebbe funzionare anche con `subfig`.

## 25.3 — Comandi per modificare i titoli

Va detto prima di tutto che caricando `tocloft` *senza* l'opzione `titles`, l'indice generale e gli elenchi di figure e tabelle *non* cominciano una nuova pagina. Se lo si desidera diventa necessario precedere i comandi `\tableofcontents`, `\listoffigures` e `\listoftables` con `\cleardoublepage`.

`\cftmark<ext>`

Sono comandi simili a `\markboth` e `\markright` e servono a stabilire l'indicatore relativo all'elenco in questione che, di solito, va nella testatina. Probabilmente non va usato.

`\cftbefore<ext>titleskip`  
`\cftafter<ext>titleskip`

Sono le dimensioni dello spazio bianco che precede e segue il titolo, vanno cambiate con `\setlength`.

`\cft<ext>titlefont`

Contiene la specificazione per il tipo di carattere da usare per il titolo; va cambiato con `\renewcommand`, per esempio con

```
\renewcommand{\cfttoctitlefont}{\Large\bfseries}
```

`\cftafter<ext>title`

Contiene codice da usare dopo il titolo; se, per esempio, si volesse che il titolo dell'elenco delle figure fosse centrato e in corsivo a corpo normale, si dovrebbe scrivere

```
\renewcommand{\cftlofttitlefont}{\hfill\itshape}
\renewcommand{\cftafterlofttitle}{\hfill}
```

Con `\renewcommand{\cftafterlottitle}{\thispagestyle{empty}}` si impone alla pagina in cui compare il titolo dell'elenco delle figure lo stile di pagina `empty`.

`\contentsname`  
`\listfigurename`  
`\listtablename`

Non sono comandi specifici di `tocloft`, ma è utile menzionarli perché contengono il titolo che può essere cambiato. Si veda anche il capitolo su `babel`.

`\tocloftpagestyle`

Definisce lo stile di pagina usato nella pagina iniziale degli indici, normalmente è `plain`. È una dichiarazione simile a `\thispagestyle`.

<sup>1</sup>Va data anche se si usa `fncychap`, che però trovo un pacchetto deplorabile.

### 25.4 — Comandi per la composizione degli indici

```
\cftdot
\cftdotsep
\cftnodots
```

Sono tre comandi che controllano i puntini guida. Il primo contiene il simbolo (normalmente un punto); il secondo la distanza espressa come multiplo di una lunghezza fissa, quindi deve essere un numero decimale; il terzo può essere usato per dire quale sia una distanza ‘troppo grande’ affinché i puntini siano effettivamente stampati. Ne vedremo poi l’uso. I valori usuali sono equivalenti alle definizioni

```
\newcommand{\cftdot}{.}
\newcommand{\cftdotsep}{4.5}
\newcommand{\cftnodots}{10000}
```

e vanno ovviamente modificati con `\renewcommand`.

```
\cftdotfill{<testo>}
```

È un comando ausiliario per la composizione delle guide dal titolo al numero di pagina negli indici, si veda più avanti; `<testo>` sono opportune istruzioni.

```
\cftsetpnumwidth{<dimen>}
\cftsetrmarg{<dimen>}
```

Sono dichiarazioni che definiscono due lunghezze. La prima stabilisce l’ampiezza della ‘box’ nella quale viene composto il numero di pagina, la seconda stabilisce la distanza dal margine destro alla quale viene spezzato un titolo troppo lungo per stare su una riga. I valori usuali nella classe `book` a corpo 10 sono equivalenti alle dichiarazioni

```
\cftsetpnumwidth{1.55em} \cftsetrmarg{2.55em}
```

e, ovviamente, il valore dato come argomento della seconda deve essere maggiore di quello dato alla prima. Una larghezza di 1,55 em nel font normale del documento può essere insufficiente se il numero delle pagine è superiore a 99, perché il numero di pagina dei capitoli è stampato in nero e, nel font standard di  $\text{\LaTeX}$ , tre cifre in neretto occupano 17,25 pt, mentre 1,55 em valgono 15 pt. Se fosse necessario modificare in questo senso il valore da usare, si possono evitare i conti scrivendo nel preambolo

```
\AtBeginDocument{\settoheight{\dimen0 }{\textbf{000}}
\edef\cftsetpnumwidth{\the\dimen0 }
\addtolength{\dimen0 }{1em}
\edef\cftsetrmarg{\the\dimen0 }}
```

È un trucco che, in una guida come questa, non può essere spiegato nel dettaglio: se vi fidate, usatelo.

```
\cftparskip
```

Questa lunghezza (elastica) contiene il valore dello spazio verticale aggiuntivo prima di ogni riga dell’indice. Normalmente vale zero, lo si può modificare con `\setlength`; non vale la pena usarlo.

```
\cftbefore<cnt>skip
\cft<cnt>indent
\cft<cnt>numwidth
```

Sono tre lunghezze relative a ciascun `<cnt>`; la prima controlla la spaziatura verticale prima della riga corrispondente, la seconda il rientro rispetto al margine sinistro, la terza la larghezza da riservare al numero. Possiamo dare qui una possibile risposta al problema menzionato all’inizio della discussione:

```
\AtBeginDocument{\addtolength{\cftsecnumwidth}{.5em}}
```

Infatti ciò che ci serve è aumentare lo spazio riservato ai numeri di una sezione per farci stare una cifra in più; la larghezza delle cifre nei tipi di carattere più comuni è mezzo em. In caso di dubbio, si può usare il sistema più complicato

```
\AtBeginDocument{\settowidth{\dimen0 }{0}
\addtolength{\cftsecnumwidth}{\dimen0 }}
```

Meglio usare `\AtBeginDocument` perché così i comandi dati come argomento vengono eseguiti quando  $\text{\LaTeX}$  ha sicuramente impostato il font principale del documento. Con `\calc` si può scrivere più semplicemente

```
\AtBeginDocument{\addtolength{\cftsecnumwidth}{\widthof{0}}}
```

```
\cftsetindent{<cnt>}{<dimen1>}{<dimen2>}
```

Scrivere `\cftsetindents{chap}{0em}{1.55em}` è equivalente a

```
\setlength{\cftchapindent}{0em}
\setlength{\cftchappnum}{1.55em}
```

Si può usare ovviamente qualsiasi `<cnt>`, ma anche il nome per esteso del contatore.

```
\cft<cnt>font
```

Sono i contenitori del tipo di carattere da usare per comporre il titolo e, se c'è, il numero. I valori usuali per la classe `book` sono equivalenti a

```
\newcommand{\cftchapfont}{\bfseries}
\newcommand{\cftsecfont}{\normalfont}
```

Vanno cambiati con `\renewcommand`.

```
\cft<cnt>presnum
\cft<cnt>aftersnum
\cft<cnt>aftersnumb
```

Sono contenitori di codice da eseguire in vari momenti. Va ricordato che il numero (se c'è) viene composto in una 'box' di ampiezza `\cft<cnt>numwidth`; di fatto il codice eseguito per comporre il numero è

```
\makebox[\cft<cnt>numwidth][l]
{\cft<cnt>presnum<numero>\cft<cnt>aftersnum}
```

A quel codice segue quello contenuto in `\cft<cnt>aftersnumb` e quindi il titolo. Per esempio, se vogliamo che un titolo di sezione compaia nella forma

(1.2) — Titolo della sezione

possiamo dire, dopo aver dato il valore corretto alla dimensione `\cftsecnumwidth`,

```
\renewcommand{\cftsecpresnum}{\hfill{}}
\renewcommand{\cftsecaftersnum}{}}
\renewcommand{\cftsecaftersnumb}{ --- }
```

Il motivo di `\hfill` è che, in questo caso, il numero dovrebbe essere composto in modo da finire sul margine destro della 'box'. Si tratta solo di un esempio e non di una buona idea, naturalmente.

Il contenuto di questi comandi è, usualmente, vuoto; va modificato con `\renewcommand`. I comandi `\cftpartaftersnum` e `\cftpartaftersnumb`, nelle classi `book`, `report` e `article` sono ignorati.

**Tabella 25.2** Valori normali di `\cft<cnt>leader` e `\cft<cnt>dotsep`

Comando	Definizione
<code>\cftpartleader</code>	<code>\large\bfseries\cftdotfill{\cftpartdotsep}</code>
<code>\cftchapleader</code>	<code>\bfseries\cftdotfill{\cftchapdotsep}</code>
<code>\cftsecleader</code>	<code>\normalfont\cftdotfill{\cftsecdotsep}</code>
<code>\cftsubsecleader</code>	<code>\normalfont\cftdotfill{\cftsubsecdotsep}</code>
<code>\cftsubsubsecleader</code>	<code>\normalfont\cftdotfill{\cftsubsubsecdotsep}</code>
<code>\cfttableader</code>	<code>\normalfont\cftdotfill{\cfttabdotsep}</code>
<code>\cftfigleader</code>	<code>\normalfont\cftdotfill{\cftfigdotsep}</code>
<code>\cftpartdotsep</code>	<code>\cftnodots</code>
<code>\cftchapdotsep</code>	<code>\cftnodots</code>
<code>\cftsecdotsep</code>	<code>\cftdotsep</code>
<code>\cftsubsecdotsep</code>	<code>\cftdotsep</code>
<code>\cftsubsubsecdotsep</code>	<code>\cftdotsep</code>
<code>\cfttabdotsep</code>	<code>\cftdotsep</code>
<code>\cftfigdotsep</code>	<code>\cftdotsep</code>

`\cft<cnt>leader`  
`\cft<cnt>dotsep`

Questi comandi contengono i dati per comporre le guide dal titolo al numero di pagina. Si possono modificare con `\renewcommand`; i valori normali sono nella tabella 25.2

Ciascuno dei comandi del primo gruppo usa `\cftdotfill` per comporre le guide, che sono costituite dalla ripetizione del simbolo contenuto in `\cftdot`. Dando come argomento `\cftnodots` le guide non compaiono perché i simboli sono troppo distanziati per apparire sulla pagina e dunque, con queste definizioni, le guide appaiono solo dal livello di sezione in giù; meglio usare solo una distanza fra puntini, appunto quella stabilita dando un valore a `\cftdotsep`. Per farli apparire è sufficiente allora dare a `\cft<cnt>dotsep` il valore `\cftdotsep`, per esempio

```
\renewcommand{\cftchapdotsep}{\cftdotsep}
```

`\cft<cnt>pagefont`

Contiene le specifiche per il tipo di carattere da usare nella composizione dei numeri di pagina. Si modifica con `\renewcommand`.

`\cft<cnt>afterpnum`

Contiene codice da eseguire dopo la composizione del numero di pagina; è essenziale se non si vogliono avere i numeri di pagina incolonnati, ma avvicinati al titolo.

`\cftparfillskip`

Con questo comando si può ottenere l'effetto detto prima; per avere i numeri di pagina dei capitoli non incolonnati, si può scrivere

```
\renewcommand{\cftchapleader}{}
\renewcommand{\cftchapafterpnum}{\cftparfillskip}
```

Si può aggiungere codice a `\cftchapleader` per avere una separazione, per esempio

```
\renewcommand{\cftchapleader}{\normalfont\textbullet\ }}
```

Tuttavia c'è il problema che i numeri di pagina sono composti in una 'box' di ampiezza fissa e allineati al margine destro di essa. Si può ovviare al problema solo modificando un comando interno di `tocloft`; do quindi la soluzione completa per i capitoli:

---

```

\renewcommand{\cftchapleader}{\normalfont\ \textbullet\ }}
\renewcommand{\cftchapafterpnum}{\cftparfillskip}
\renewcommand{\cftchapfillnum}[1]{%
  {\cftchapleader}\nobreak
  \mbox{\cftchappagefont #1}\cftchapafterpnum\par}

```

e ripetere in modo analogo per tutti i comandi della forma `\cft<cnt>fillnum`; naturalmente, a questo punto sarebbe molto più semplice usare una sola definizione per ciascun tipo, secondo il seguente modello:

```

\renewcommand{\cftchapfillnum}[1]{%
  \mbox{\normalfont\ \textbullet\ }\nobreak
  \mbox{\cftchappagefont #1}\cftparfillskip\par}

```

### 25.5 — Nuovi elenchi

Il pacchetto `tocloft` permette anche di definire nuovi elenchi, purché l'utente si definisca gli ambienti adeguati per compilarli. Ci sono altri pacchetti che hanno funzioni analoghe, in particolare `float` e `ntheorem`. Ci interessa qui assicurarci la compatibilità con `float`: se si adopera `tocloft` per interventi significativi sulla composizione degli elenchi, è importante che si possano eseguire su quelli definiti tramite `float`.

```
\newlistof[<contatore>]{<contatore_0>}{<ext>}{<testo>}
```

Questo comando definisce tutta la strumentazione per modificare la composizione di un nuovo elenco secondo le modalità descritte in precedenza, dove al posto di `<cnt>` si usa `<contatore>`; questo contatore è legato a `<contatore_0>`, cioè viene azzerato a ogni scatto di quest'ultimo. Per esempio, dopo

```
\newlistof[chapter]{algorithm}{loa}{Elenco degli algoritmi}
```

avremo a disposizione un contatore `algorithm`, che viene azzerato a ogni scatto di `chapter`, e comandi come `\cftalgorithmpresnum` o `\cftalgorithmnumwidth`. Ciò che il pacchetto non fornisce sono gli ambienti che permettano di compilare questo elenco. Si può usare allo scopo `float`, a patto di trascurare il comando `\listof` a favore di quello introdotto da `tocloft`.

```
\listof<contatore>
```

Una volta definito un nuovo contatore con il comando precedente, abbiamo a disposizione il comando per compilare l'elenco corrispondente. Proseguendo nell'esempio precedente, il comando sarà `\listofalgorithm`.

### 25.6 — Compatibilità con float

Supponiamo di aver costruito con `float` un nuovo ambiente galleggiante:

```

\newfloat{algorithm}{htp}{loa}[chapter]
\floatname{algorithm}{Algoritmo}

```

Potremmo compilare l'elenco di questi ambienti tramite il comando

```
\listof{algorithm}{Elenco degli algoritmi}
```

ma non potremmo usare le funzioni di `tocloft`. Il trucco è di combinare nel modo opportuno i comandi:

```

\newlistof[chapter]{algorithm}{loa}{Elenco degli algoritmi}
\newfloat{algorithm}{htp}{loa}[chapter]
\floatname{algorithm}{Algoritmo}

```



e di adoperare `\listofalgorithm` per generare l'elenco, eventualmente dopo aver modificato i parametri relativi con i comandi di `tocloft`. Unica noia è un messaggio di avviso come

```
Package float Warning: Can't redefine counter variable for algorithm
```

Lo si potrebbe evitare con una ridefinizione del comando `\newfloat` o qualche bieco trucco; tuttavia il messaggio è del tutto innocuo e la faccenda va a buon fine senza alcun problema, perché il file ausiliario scritto da `float` (in questo caso con estensione `.loa`) è in una forma perfettamente comprensibile da `tocloft`.

È però importante, per la compatibilità, non approfittare della possibilità data da `float` di dichiarare nuovi ambienti all'interno del documento. Li si dichiarino sempre e solo nel preambolo.

### 25.7 — Compatibilità con altri pacchetti

Il pacchetto `tocloft` è compatibile con `tocbibind` che si occupa di inserire automaticamente nell'indice generale le voci relative alla bibliografia e all'indice analitico. Ci sono alcuni problemini con certe funzionalità secondarie, si consultino la documentazione ufficiale.

C'è buona compatibilità anche con `minitoc`, sebbene ci possa essere qualche lieve problema di scelta di caratteri. Le possibili soluzioni sono riportate nella documentazione ufficiale.

**graphicx**

```
\usepackage[<driver>,<opzione>,...]{graphicx}
```

Il pacchetto `graphicx` è il metodo standard per l'inclusione di oggetti grafici esterni in documenti  $\text{\LaTeX}$  tramite il comando `\includegraphics`, ma anche per creare effetti grafici di rotazione, ingrandimento o riduzione di normale testo.

**26.1 — Opzioni**

Le opzioni a `graphicx` sono di due tipi: specifica del driver e opzioni propriamente dette. Occorre specificare un driver perché il formato DVI si appoggia a uno di essi per la stampa finale, ma vedremo che nella maggior parte dei casi non sarà necessario scrivere esplicitamente l'opzione. I driver conosciuti sono

<code>dvips</code>	<code>xetex</code>	<code>dviwin</code>	<code>pctexhp</code>
<code>xdvi</code>	<code>pdftex</code>	<code>oztex</code>	<code>pctex32</code>
<code>dvipdf</code>	<code>dvipsone</code>	<code>textures</code>	<code>truetestex</code>
<code>dvipdfm</code>	<code>dviwindo</code>	<code>pctexps</code>	<code>tcidvi</code>
<code>dvipdfmx</code>	<code>emtex</code>	<code>pctexwin</code>	<code>vtex</code>

ma nella maggior parte dei casi si adopera `dvips`, `pdftex` oppure `xetex` e il pacchetto ha la capacità di scoprire automaticamente se deve caricare uno di questi driver particolari. In effetti la compilazione di un documento  $\text{\LaTeX}$  può avvenire tramite la chiamata di `latex`, `pdflatex` oppure `xelatex` (con metodi che differiscono da un sistema operativo a un altro, ma questo è irrilevante). In questi casi la semplice chiamata `\usepackage{graphicx}` è sufficiente.

Solo nel caso si debba usare `dvipdfmx` per la trasformazione del DVI in PDF sarà necessario indicare esplicitamente l'opzione.

Descriviamo ora le altre opzioni al pacchetto.

`draft` | `final`

Con una di queste opzioni si può eseguire una scelta diversa da quella data come opzione globale in `\documentclass`. Per esempio, se si dà l'opzione globale `draft`, che serve in generale per mostrare chiaramente gli 'overfull', il pacchetto `graphicx` non include i documenti grafici, ma mostra solo un rettangolo vuoto con il nome del file esterno richiesto. Se invece si specifica l'opzione `final` al pacchetto, verranno inclusi i documenti grafici. Nella versione finale del documento non si specificherà alcuna di queste opzioni, né globalmente né al pacchetto.

`hiderotate`

Il pacchetto `graphicx` non cerca di fare in modo che siano mostrate le rotazioni di oggetti grafici; è utile nel caso in cui il programma di visualizzazione sia carente in questo aspetto.

`hidescale`

Vale il discorso per `hiderotate`, ma riferito a ingrandimenti e riduzioni di oggetti grafici.

`hiresbb`

Un file PostScript può contenere informazioni sulla 'bounding box' più accurate rispetto alle normali dimensioni in punti; con questa opzione si chiede di usare queste informazioni.

demo

Non viene incluso alcun documento grafico esterno, al cui posto viene mostrato solo un rettangolo vuoto di dimensioni fisse.

## 26.2 — Documenti grafici esterni

L'esistenza di formati grafici diversi è spesso causa di violenti mal di testa ai principianti che non sanno mai quali usare e quali includere. Quelli che le varie incarnazioni di L<sup>A</sup>T<sub>E</sub>X possono includere sono:

- 'Encapsulated PostScript' (estensione usuale `.eps`),
- 'Portable Document Format' (estensione usuale `.pdf`),
- 'Portable Network Graphic' (estensione usuale `.png`),
- 'Joint Photographic Expert Group' (estensione usuale `.jpg`),
- 'Meta PostScript' (estensione usuale `.mps`).

L'ultimo tipo, forse sconosciuto ai più, comprende i documenti PostScript generati da METAPOST.

Va subito detto che i vari driver non riescono a comprenderle tutte, in generale; i driver che li possono importare tutti (e in realtà anche altri) sono `dvipdfmx` e `xetex`. Non entreremo nei dettagli di formati più esoterici, né nel problema di quale formato sia più adatto di un altro: per questo occorrerebbe molto più spazio. Ricordiamo solo che 'Encapsulated PostScript' è una forma ristretta di PostScript; si faccia attenzione a chiedere questo tipo di file quando si esporta qualcosa da un altro programma in questo formato. Il formato MPS è ancora più ristretto di questo.

Detto che con `xelatex` si possono importare tutti i quattro formati principali, va osservato che `latex` (con il driver `dvips`) può includere solo EPS, mentre `pdflatex` (che ammette solo il driver `pdftex`) può includere PNG, JPEG, PDF e MPS.

Non è un problema il fatto che `pdflatex` non comprenda il formato EPS, perché è molto facile eseguire la conversione in PDF. Un po' meno facile è con `latex`: convertire da PDF in EPS è banale, un po' più complicato passare da PNG e JPEG in EPS. Ci sono molti programmi di conversione, qualcuno più, qualcuno meno efficiente.

Una parola sul formato MPS: spesso i file prodotti da METAPOST hanno estensione numerica (`pippo.1`, `pippo.2`, eccetera). Si consiglia di modificare l'estensione in `.mps`; le ultime versioni di METAPOST hanno introdotto il comando `filenamememplate` che permette di definire file di output con estensione arbitraria e il numero nel nome.

## 26.3 — Comandi

I comandi di `graphicx` hanno quasi tutti un argomento opzionale che comprende una lista nella forma `chiave=valore`, che elencheremo subito dopo la sintassi del comando.

`\rotatebox[<opzioni>]{<angolo>}{<testo>}`

Il `<testo>` viene ruotato dell'angolo `{<angolo>}` espresso in gradi, in senso antiorario se il numero è positivo. Il `<testo>` è codice qualsiasi che abbia senso in una `\mbox`.

Opzioni: 

`origin=<pos>`  
`x=<dimen>`  
`y=<dimen>`  
`units=<numero>`

È possibile specificare l'origine della rotazione specificando `origin=` seguito da uno o due caratteri nella lista `lrctbB`; se non si specifica nulla, l'origine è il punto di riferimento della 'box' che si avrebbe con `\mbox{<testo>}`. I caratteri servono a riferirsi a 'sinistra' (`l`), 'destra' (`r`), 'centro' (`c`), 'alto' (`t`), 'basso' (`b`) e 'linea di base' (`B`).

Per esempio, con `origin=c` la rotazione è rispetto al centro geometrico della 'box'; con `origin=tr` la rotazione è rispetto all'angolo in alto a destra. Il carattere `B` si riferisce alla

linea di base (una ‘box’ può estendersi anche sotto la linea immaginaria lungo la quale si allineano i caratteri di stampa).

L’origine può anche essere specificata dando due coordinate (con la dimensione), per esempio `x=2mm,y=3mm`; le coordinate rappresentano lo spostamento rispetto al punto di riferimento della ‘box’, che è il punto sul margine sinistro e sulla linea di base.

L’opzione `units` serve a specificare una unità di misura degli angoli; per esempio con `units=-360` si rovescia il senso di rotazione, mentre con `units=6.283185` si usano angoli in radianti.

```
\scalebox{⟨numero_x⟩}[⟨numero_y⟩]{⟨testo⟩}
```

Il `⟨testo⟩` viene scalato in senso orizzontale usando il rapporto di ingrandimento `⟨numero_x⟩` e in senso verticale con il rapporto di ingrandimento `⟨numero_y⟩`. Se non viene espresso l’argomento opzionale, si considera `⟨numero_y⟩ = ⟨numero_x⟩`. Questi due rapporti possono essere numeri decimali qualsiasi, anche negativi (che corrispondono a una riflessione).

```
\reflectbox{⟨testo⟩}
```


Equivale a `\scalebox{1}[-1]{⟨testo⟩}`.

```
\resizebox{⟨dimen_x⟩}{⟨dimen_y⟩}{⟨testo⟩}
\resizebox*{⟨dimen_x⟩}{⟨dimen_y⟩}{⟨testo⟩}
```

Il `⟨testo⟩` viene ingrandito o ridotto in modo da essere largo `⟨dimen_x⟩` e alto (sopra la linea di base) `⟨dimen_y⟩`. Con la forma variata `\resizebox*` è l’altezza globale (compresa quindi la profondità, cioè quanto va sotto la linea di base) a essere portata a valere `⟨dimen_y⟩`. Si veda la differenza:

`\resizebox{1cm}{1cm}{g}` →  `\resizebox*{1cm}{1cm}{g}` → 

Nei primi due argomenti si possono usare `\width`, `\height` e `\depth`, che si riferiscono alle dimensioni naturali di `\mbox{⟨testo⟩}`. Perciò, per scalare solo in verticale si può usare `\resizebox{⟨width⟩}{1cm}{g}`, per avere un ingrandimento solo orizzontale si può dare `\resizebox{1cm}{⟨height⟩}{g}`. In uno solo dei due argomenti si può dare il carattere speciale `!`, che indica a  $\text{\LaTeX}$  di scalare il `⟨testo⟩` secondo l’ingrandimento richiesto e senza operare distorsioni:

`\resizebox{1cm}{!}{g}` → 

```
\includegraphics[⟨opzioni⟩]{⟨file⟩}
```

È il comando più usato, serve per inserire nel documento  $\text{\LaTeX}$  un `⟨file⟩` grafico esterno. Questo crea un oggetto simile a un carattere, di altezza e larghezza uguali a quelle del contenuto del file grafico, se non ne sono state specificate altre con le opzioni. Lo si può usare ovunque, con la sola precauzione di proteggerlo (`\protect`) se dovesse essere inserito in un argomento mobile: è una falsa credenza che questo comando possa andare solo in un ambiente `figure`. Potrebbe benissimo essere adoperato, invece, per costruirsi un simbolo particolare che non sia disponibile più facilmente.

Si consiglia di dare solo il nome proprio del `⟨file⟩`, tralasciando l’estensione: in questo modo  $\text{\LaTeX}$  (o meglio, il driver) andrà in cerca dell’estensione corretta. Per esempio, se sono disponibili i file `pippo.eps` e `pippo.pdf`, potremo compilare sia con `latex` che con `pdflatex` senza apportare modifiche al nostro documento, ammesso che i due file grafici siano equivalenti, con il comando

```
\includegraphics{pippo}
```

L'ordine di precedenza usato da `latex` è: prima `.eps` e poi `.ps`. Quello di `pdflatex` è

```
.png .pdf .jpg .mps .jpeg .jbig2 .jb2
.PNG .PDF .JPG .JPEG .JBIG2 .JB2
```

(qui compare il formato `JBIG2` di cui però non parleremo più). Ciò significa che se sono disponibili i file `pippo.png` e `pippo.pdf` verrà preso il primo, a meno che non si specifichi l'estensione.

Discutiamo ora le chiavi che si possono specificare nell'argomento opzionale, che servono a modificare il modo di inserire l'oggetto grafico. Le divideremo in vari gruppi; alcune di esse sono 'booleane', e in questo caso non occorre dare `=true` per abilitarle, ma basta specificare la chiave (per esempio `hiresbb` o `crop`).

Una parola sulle dimensioni: possono essere date senza unità di misura e, in questo caso, è implicita l'unità `TeX bp`: `72 bp` è uguale a un pollice, cioè `2,54 cm`. Possono anche essere date con l'unità di misura esplicita, che può essere una qualunque di quelle comprese da `TeX`.

Opzioni: 

<code>bb=&lt;dimen<sub>1</sub>&gt; &lt;dimen<sub>2</sub>&gt; &lt;dimen<sub>3</sub>&gt; &lt;dimen<sub>4</sub>&gt;</code>
<code>bbllx=&lt;dimen&gt;</code>
<code>bbly=&lt;dimen&gt;</code>
<code>bburx=&lt;dimen&gt;</code>
<code>bbury=&lt;dimen&gt;</code>

Queste opzioni correggono (o specificano) la cosiddetta 'bounding box' dell'immagine, cioè le dimensioni del minimo rettangolo che la contiene. Il linguaggio PostScript usa specificare quattro dimensioni, che sono in ordine le coordinate del vertice in basso a sinistra e quelle del vertice in alto a sinistra. Si possono specificare tutte e quattro insieme con `bb=` oppure una alla volta con le quattro opzioni seguenti, i cui nomi dovrebbero essere chiari.

Si noti che questo equivale a una traslazione del sistema di riferimento rispetto a quello specificato nel file, se è un EPS oppure PDF. Per gli altri formati grafici questa chiave è inutile. Si veda più avanti l'opzione `clip`.

Opzioni: 

<code>hiresbb</code>
----------------------

Questa opzione ha senso solo per file di tipo EPS e MPS. La tradizionale 'bounding box' dei file PostScript è espressa in 'punti PostScript', corrispondenti all'unità `TeX bp`. Quando si compila con `pdflatex`, i file di tipo PNG, PDF e JPEG vengono inclusi con dimensioni accurate; questo può non essere vero per gli altri due tipi; naturalmente è necessario che il file PostScript che si include abbia l'informazione perché l'effetto sia apprezzabile. Si veda uno degli esempi dove la differenza è sensibile.

Opzioni: 

<code>viewport=&lt;dimen<sub>1</sub>&gt; &lt;dimen<sub>2</sub>&gt; &lt;dimen<sub>3</sub>&gt; &lt;dimen<sub>4</sub>&gt;</code>
<code>trim=&lt;dimen<sub>1</sub>&gt; &lt;dimen<sub>2</sub>&gt; &lt;dimen<sub>3</sub>&gt; &lt;dimen<sub>4</sub>&gt;</code>

Con `viewport` si apre una 'finestra' rettangolare sull'immagine; le quattro dimensioni sono le coordinate del vertice in basso a sinistra e in alto a destra di questo rettangolo, rispetto al sistema di riferimento naturale dell'immagine. Con `trim` si dice a `TeX` di ritagliare l'immagine di quelle quattro dimensioni, andando in senso antiorario dal lato sinistro: sinistra, sotto, destra, alto. Si veda più avanti l'opzione `clip`.

Opzioni: 

<code>angle=&lt;numero&gt;</code>
<code>origin=&lt;pos&gt;</code>

Il `<numero>` dato come valore a `angle` indica la rotazione richiesta, in senso antiorario, rispetto al punto di riferimento che è il vertice in basso a sinistra della 'bounding box'. Si può modificare la scelta di questo punto usando una sintassi simile a quella vista per `\rotatebox` con la chiave `origin`.

Opzioni: 

<code>width=&lt;dimen&gt;</code>
<code>height=&lt;dimen&gt;</code>
<code>totalheight=&lt;dimen&gt;</code>
<code>keepaspectratio</code>
<code>scale=&lt;numero&gt;</code>

Le dimensioni finali dell'immagine sono ingrandite o ridotte secondo quanto specificato con i valori a `width` e `height`. Dandone una sola, l'altra verrà calcolata in modo che non ci siano distorsioni; si possono dare entrambe e non avere distorsioni specificando la chiave booleana `keepaspectratio`, in questo caso verrà scelto l'ingrandimento o riduzione tale che non si ecceda alcuna delle dimensioni specificate.

Si ricorra a `totalheight` quando si sia eseguita una rotazione che porta l'immagine, agli occhi di  $\TeX$ , ad avere una profondità, cioè a estendersi sotto la linea di base. Si faccia anche attenzione all'ordine con cui vengono lette le opzioni, lo rivedremo con gli esempi.

Opzioni: 

<code>clip</code>
-------------------

È un'opzione booleana da usare in connessione con `bb`, `viewport` e `trim`. Con questa opzione si fa in modo che l'immagine sia effettivamente ritagliata secondo le indicazioni. Capita abbastanza spesso che un'immagine abbia bordi bianchi troppo grandi o che se ne desideri solo una parte (rettangolare). Possiamo misurare quanto si vuole tagliare scrivendo

```
\begin{center}
\setlength{\fboxsep}{0pt}
\fbox{\includegraphics{pippo}}
\end{center}
```

che creerà una cornice attorno all'immagine. Una volta stabilito quanto si deve ritagliare, diciamo 1 cm a sinistra, 2 cm in basso, niente a destra e niente in alto, potremmo scrivere

```
\includegraphics[trim=1cm 2cm 0 0,clip]{pippo}
```

Equivalente a specificare l'opzione `clip` è usare il comando `\includegraphics*` con le stesse opzioni (tranne `clip`). Se non si specifica l'opzione `clip`, l'immagine occuperà solo lo spazio indicato dalle altre opzioni, ma si vedrebbe anche ciò che esce dalla 'bounding box' così ottenuta.

Opzioni: 

<code>draft</code>
--------------------

Modifica per la sola immagine in questione la scelta dell'opzione `final` data globalmente o al pacchetto `graphicx`.

Opzioni: 

<code>type=&lt;tipo&gt;</code>
<code>ext=&lt;ext<sub>1</sub></code>
<code>read=&lt;ext<sub>2</sub></code>
<code>command=&lt;comando&gt;</code>

Con queste opzioni si può cercare di far leggere un file grafico con estensioni strane. Per esempio, se si ha un file prodotto da `METAPOST` con estensione `.1`, per esempio `pippo.1`, e lo si vuole includere con `pdflatex`, si può scrivere

```
\includegraphics[type=mps,ext=.1,read=.1]{pippo}
```

Diventa piuttosto complicato scrivere tutto questo per ogni file di questo tipo, se ricordiamo che normalmente `METAPOST` ha come uscita file con estensioni numeriche; in alcuni casi, in presenza di molti file di questo tipo e con varie estensioni numeriche, potremmo definire un comando personale

```
\newcommand{\includemps}[3][ ]{%
\includegraphics[type=mps,ext=.#3,read=.#3,#1]{#2}}
```

e includere il nostro `pippo.1` scalato alla metà con

```
\includemps[scale=0.5]{pippo}{1}
```

Si noti che `type=meps` funziona con `pdflatex`, mentre con `latex` occorre scrivere `type=eps`.

Si potrebbe ottenere una sintassi più amichevole

```
\includemps[⟨opzioni⟩]{⟨nome⟩.⟨ext⟩}
```

cioè del tipo `\includemps[scale=0.5]{pippo.1}` con un sistema più complesso:

```
\usepackage{graphicx,ifpdf}
\makeatletter
\edef\mps@or@eps{\ifpdf m\else e\fi ps}
\newcommand{\includemps}[2][{}]{%
  \filename@parse{#2}%
  \edef\ig@opt{type=\mps@or@eps,ext=. \filename@ext,read=. \filename@ext}%
  \expandafter\includegraphics\expandafter
  [\ig@opt,#1]{\filename@area\filename@base}}
\makeatother
```

che funzionerebbe sia con `latex` che con `pdflatex`. Ma non ammette la ricerca nelle sottocartelle specificate con `\graphicspath`.

```
\graphicspath{⟨lista⟩}
```

Supponiamo che per motivi di ordine si desideri raggruppare i file grafici in cartelle sottostanti quella dove si trova il documento L<sup>A</sup>T<sub>E</sub>X principale. Si può evitare di scrivere il percorso esplicitamente nel `⟨nome⟩` dato come argomento di `\includegraphics` usando questo comando. La sintassi è un po' bizzarra: l'argomento `⟨lista⟩` è un elenco di nomi di sottocartelle ciascuno circondato da graffe. Perciò se i file grafici stanno nelle cartelle `pippo-png` e `pippo-jpg` (divise per formato grafico, per esempio), daremo

```
\graphicspath{{pippo-png/}{pippo-jpg/}}
```

Non si dimentichi la barra finale che dovrebbe andare bene su tutti i sistemi operativi moderni. Se c'è una sola sottocartella `pippo-grf` si deve scrivere

```
\graphicspath{{pippo-grf/}}
```

ancora con le parentesi graffe “in più”. È possibile dare percorsi più complicati, ma si eviti di risalire nella struttura del disco o di dare percorsi assoluti (che cominciano con `/`), perché spesso i driver non ammettono la ricerca di file in quel modo per motivi di sicurezza.

```
\DeclareGraphicsExtensions{⟨lista⟩}
```

La `⟨lista⟩` è un elenco di estensioni possibili dei nomi di file grafici, separati da virgole. Si può usare questo sistema per modificare l'ordine di precedenza nella ricerca dei file.

```
\DeclareGraphicsRule{⟨ext⟩}{⟨tipo⟩}{⟨file⟩}{⟨comando⟩}
```

Non descriveremo questo comando in tutti i dettagli, limitandoci a un caso particolare che può tornare utile. Il driver `dvips` non riconosce il formato MPS, che però è una forma ristretta di EPS. Possiamo intanto dichiarare che un file MPS è un EPS e che i dati della ‘bounding box’ possono essere letti direttamente da quel file:

```
\DeclareGraphicsRule{.mps}{eps}{.mps}{}
```

Rimane però il problema di aggiungere `.mps` alla lista di estensioni riconosciute dal driver. Un modo alquanto magico è di scrivere il seguente incantesimo nel preambolo:

```

\usepackage{graphicx,ifpdf} % pacchetti necessari
\ifpdf\else
  \makeatletter
  \edef\Gin@extensions{\Gin@extensions,.mps}
  \makeatother
  \DeclareGraphicsRule{.mps}{eps}{.mps}{}
\fi

```

Il terzo argomento di `\DeclareGraphicsRule` serve a dire quale file contiene i dati della ‘bounding box’: al nome (senza estensione) del file si aggiunge la stringa di caratteri specificata nel terzo argomento. Siccome il file MPS contiene i dati richiesti, facciamo leggere questo. Il quarto argomento è vuoto, non lo sarebbe solo in casi molto esoterici. Si noti però che questo sistema funziona solo per file con estensione `.mps`.

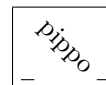
#### 26.4 — Esempi

Negli esempi si userà come file grafico `grafo.mps`, generato dal sorgente `grafo.mp`. Ogni esempio sarà circondato da una cornice oltre che preceduto e seguito da un filetto sulla linea di base, per vedere l’effetto. Si tenga conto che la cornice ha una distanza di 2pt da ciascun lato della figura.

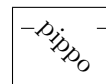
1. `\rotatebox{45}{pippo}`



2. `\rotatebox[origin=tr]{-45}{pippo}`



3. `\rotatebox[origin=1B]{-45}{pippo}`



4. `\rotatebox[origin=1]{-45}{pippo}`



5. `\scalebox{2}{pippo}`



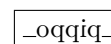
6. `\scalebox{2}[1.5]{pippo}`



7. `\scalebox{-2}[1.5]{pippo}`



8. `\reflectbox{pippo}`





9. `\resizebox{4cm}{1cm}{pippo}`



10. `\resizebox*{4cm}{1cm}{pippo}`



11. `\resizebox{4cm}{!}{pippo}`



12. `\resizebox{4cm}{\height}{pippo}`



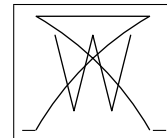
13. `\resizebox{2cm}{!}{\rotatebox{45}{pippo}}`



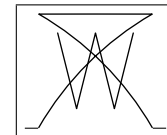
14. `\rotatebox{45}{\resizebox{2cm}{!}{pippo}}`



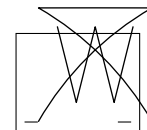
15. `\includegraphics{grafo}`



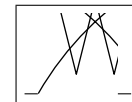
16. `\includegraphics[hiresbb]{grafo}`



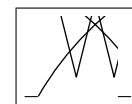
17. `\includegraphics[viewport=0 0 30 30]{grafo}`



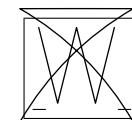
18. `\includegraphics[viewport=0 0 30 30,clip]{grafo}`



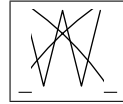
19. `\includegraphics*[viewport=0 0 30 30]{grafo}`



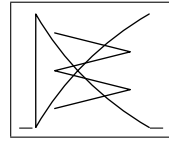
20. `\includegraphics[trim=10 5 7 8]{grafo}`



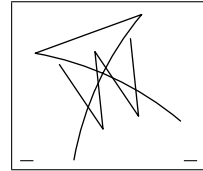
21. `\includegraphics[trim=10 5 7 8,clip]{grafo}`



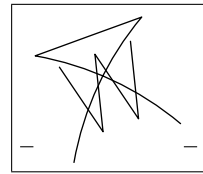
22. `\includegraphics[angle=90]{grafo}`



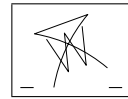
23. `\includegraphics[angle=20]{grafo}`



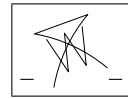
24. `\includegraphics[angle=20,origin=c]{grafo}`



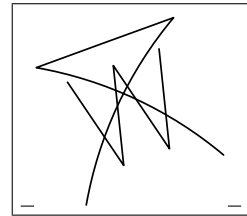
25. `\includegraphics[angle=20,scale=0.5]{grafo}`



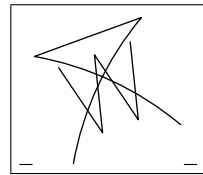
26. `\includegraphics[angle=20,origin=c,scale=0.5]{grafo}`



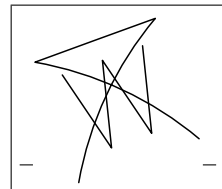
27. `\includegraphics[height=2cm,angle=20]{grafo}`



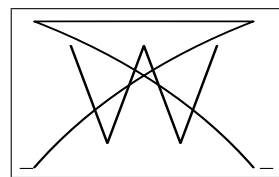
28. `\includegraphics[angle=20,height=2cm]{grafo}`



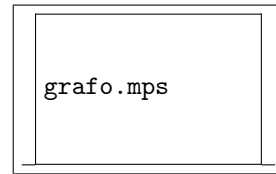
29. `\includegraphics[angle=20,origin=c,height=2cm]{grafo}`



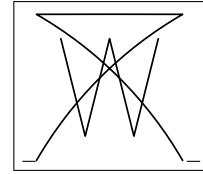
30. `\includegraphics[height=2cm,width=3cm]{grafo}`



31. `\includegraphics[height=2cm,width=3cm,draft]{grafo}`



32. `\includegraphics[height=2cm,width=3cm,keepaspectratio]{grafo}`



Si noti la differenza tra la figura 15 e la 16.

### 26.5 — Avvertenze

La lettura delle opzioni a `\includegraphics` è da sinistra a destra; si noti la differenza fra ingrandimento seguito da rotazione o viceversa (esempi 27 e 28). In certi casi succede che  $\text{\LaTeX}$  si lamenti per certe combinazioni

`height=<dimen>,angle=<numero>`

e la cura è semplice: usare `totalheight` che, del resto, è spesso la chiave giusta da adoperare. Si osservi anche la differenza tra gli esempi 17 e 18: nel primo si vede l'intera immagine che però esce dalla cornice che rappresenta lo spazio riservato da  $\text{\LaTeX}$  all'oggetto grafico; nel secondo, viceversa, l'opzione `clip` ritaglia alla grandezza richiesta.

**color**

```
\usepackage{⟨⟨opzione⟩,...⟩}{color}
```

Il pacchetto `color` permette di usare colori nei documenti  $\text{\LaTeX}$ . Esiste un altro pacchetto, molto più potente, `xcolor`, ma anche più complicato. Tuttavia i comandi di `color` sono compatibili con `xcolor` ed è meglio imparare a usare quello più semplice.

**27.1 — Opzioni**

Le opzioni a `color` sono, come quelle di `graphicx`, divise in due tipi: scelta del driver e opzioni propriamente dette. Per la scelta del driver valgono le stesse cose dette per `graphicx`, quindi rimandiamo a quel capitolo.

**monochrome**

Questa opzione disabilita tutti i comandi relativi senza però che producano errori; semplicemente non fanno nulla. Può essere indicato per le bozze.

**dvipsnames | nodvipsnames**

Il driver `dvips` definisce 68 nomi di colori e carica automaticamente l'opzione `dvipsnames`. Si può dare l'opzione contraria se non si desidera avere questi nomi a disposizione.




**usenames**

L'opzione mette a disposizione come  $\langle\text{colore}\rangle$  tutti i nomi dei colori dello spazio **named**, si veda più avanti.



**27.2 — Spazi di colore**

La scelta dei colori può essere fatta tramite quattro diverse modalità, che chiameremo *spazi di colore*. Un elemento di questi spazi è un colore; alcuni di essi possono essere espressi in più spazi, altri no.



**rgb**

Un colore in questo spazio è definito da una terna di numeri reali nell'intervallo  $[0, 1]$  (estremi inclusi). La prima componente definisce il canale rosso, la seconda il verde e la terza il blu: è ben noto che moltissimi colori possono essere espressi come miscuglio di questi tre colori primari. Il nome dello spazio deriva appunto da 'red', 'green', 'blue'. Per esempio, il colore rosso puro è definito tramite la terna  $(1, 0, 0)$ , il verde con  $(0, 1, 0)$  e il blu con  $(0, 0, 1)$ . Con  $(0.5, 0.5, 0.5)$  si otterrà una tonalità di grigio , con  $(1, 1, 0)$  un giallo puro , con  $(0.8, 0.8, 0)$  una tonalità di giallo .

**cmyk**





Lo spazio **cmyk** è più adatto di **rgb** alla stampa, ma non c'è da preoccuparsi perché internamente i colori vengono tradotti in un unico spazio. Il numero di colori disponibili è enormemente superiore, perché i colori sono espressi tramite una quaterna di numeri nell'intervallo  $[0, 1]$ . Le componenti denotano rispettivamente la quantità di ciano (azzurro, cyan) che in **rgb** è  $(0, 1, 1)$ , di magenta che in **rgb** è  $(1, 0, 1)$ , di giallo (yellow) che in **rgb** è  $(1, 1, 0)$  e di nero (la 'k' sta per 'black'). Per esempio, il colore corrispondente a  $(0.1, 0.1, 0.9, 0.1)$  è , mentre quello corrispondente a  $(0.1, 0.1, 0.9, 0.3)$  è .



**gray**

Questo è il più facile: un colore è espresso da un numero nell'intervallo  $[0, 1]$  che rappresenta una tonalità di grigio; con 0 si ottiene il nero , con 0.5 un grigio già visto  e con 1 il bianco.

**named**






In questo spazio i colori hanno un nome che deve essere noto al driver (con l'opzione `dvipnames`). Ricordarsi di dare l'opzione `usenames`, in questo caso. La tabella dei nomi con i colori corrispondenti è nella sezione 27.6.



Proviamo a dare qualche nozione alla buona sugli spazi di colori. Lo spazio **rgb** usa un metodo *sottrattivo*; la terna  $(1, 0, 0)$  significa assenza di verde e blu, quindi otteniamo il rosso . La terna  $(0, 0.5, 0)$  significa assenza di rosso e blu, con una quantità media di verde  che possiamo confrontare con il verde puro  $(0, 1, 0)$  . Siccome il metodo è sottrattivo, più piccoli sono i valori, più scura è la tonalità, fino al nero che è l'assenza di colori  $(0, 0, 0)$  .

Lo spazio **gray** è un sottospazio di **rgb**: specificare il numero  $r$  equivale a specificare la terna  $(r, r, r)$  nello spazio **rgb**; quindi 0 dà , mentre 0.7 dà : più vicino è il numero a 1 più il grigio è chiaro.

Lo spazio **cmymk** è invece *additivo*, quindi il bianco è specificato con la quaterna  $(0, 0, 0, 0)$ :








```
\fcolorbox[cmymk]{0,0,0,1}{0,0,0,0}{pippo}
```

produce . Vediamo l'effetto delle quaterne  $(0, 0, 0, 1)$  ,  $(0.5, 0, 0, 0)$  ,  $(0.5, 0, 0, 0.5)$   e  $(0.5, 0, 0, 1)$  .

Come si vede, specificando 1 nella quarta componente si ottiene sempre il nero, perché lo si aggiunge al grado massimo. La quaterna  $(1, 1, 0, 0)$  significa aggiungere ciano e magenta al massimo , che non è blu. Così, con  $(1, 1, 1, 0)$   non si ha nero puro.

**27.3 — Definire colori**

In generale non è bene specificare i colori direttamente con i valori numerici, è molto meglio definire nomi astratti in modo da poterli modificare, al bisogno, agendo solo sulla loro definizione. Alcuni colori sono universali e hanno già un nome predefinito sempre disponibile:

black  red  green  blue   
cyan  magenta  yellow  white

```
\definecolor{<colore>}{<spazio>}{<specifica>}
```

Il  $\langle colore \rangle$  è una stringa alfanumerica arbitraria; con  $\langle spazio \rangle$  intendiamo uno dei quattro nomi elencati prima, cioè **rgb**, **cmymk**, **gray** o **named**. La  $\langle specifica \rangle$  è una terna di numeri separati da virgole (si deve usare il punto per separare la parte decimale) nel caso di **rgb**; una quaterna nel caso di **cmymk**; un solo numero nel caso di **gray**; un nome nel caso di **named**. Per esempio potremmo definire

```
\definecolor{light-blue}{rgb}{0.8,0.85,1}
\definecolor{mygrey}{gray}{0.75}
\definecolor{GY}{named}{GreenYellow}
```

e questi nomi potranno essere usati negli argomenti dei comandi che vedremo in seguito. Qual è il vantaggio nell'ultima definizione? Che usando **GY** nel documento ci basterà modificare la riga di definizione per cambiare tutte le parti dove abbiamo richiesto questo colore 'astratto' e potremmo anche usare uno spazio diverso. Attenzione: i colori predefiniti non appartengono allo spazio di colore **named** che li nomina con l'iniziale maiuscola. Perciò delle due righe

```
\definecolor{nero}{named}{black}
\definecolor{nero}{named}{Black}
```

la prima dà un errore, mentre la seconda definisce un alias per il colore 'Black'. Basterebbe però scrivere

```
\definecolor{nero}{gray}{0}
```

per non avere problemi.

#### 27.4 — Colorare il testo

```
\color{<colore>}
\color[<spazio>]{<specifica>}
\textcolor{<colore>}{<testo>}
\textcolor[<spazio>]{<specifica>}{<testo>}
```

La differenza fra `\color` e `\textcolor` è la stessa che intercorre fra, per esempio, `\bfseries` e `\textbf`: il primo comando è una dichiarazione il cui effetto termina quando si chiude il gruppo in cui compare, il secondo invece ha come argomento un `<testo>`; in entrambi i casi lo scopo è di colorare il testo.

Senza l'argomento opzionale i due comandi accettano come primo (o unico) argomento un nome di colore: predefinito oppure introdotto tramite `\definecolor`.

L'argomento opzionale è uno fra **rgb**, **cmymk**, **gray** o **named**, in questo caso l'argomento obbligatorio che segue è una specifica secondo le regole dello spazio scelto (una terna, una quaterna, un solo numero o un nome). Nel caso in cui si sia data l'opzione `usenames`, i nomi appartenenti allo spazio **named** possono essere usati direttamente come `<colore>`.

#### 27.5 — Colorare sfondi

```
\pagecolor{<colore>}
\pagecolor[<spazio>]{<specifica>}
```

La sintassi è simile a quella per `\color` per quanto riguarda gli argomenti. L'effetto del comando è di modificare il colore di sfondo della pagina, ma occorre ricordare che si tratta di una dichiarazione *globale* e quindi non basta chiuderla in un gruppo per annullarla. Si può però ricorrere al pacchetto `afterpage`, dando i seguenti comandi in un punto che sappiamo cadrà nella pagina che vogliamo con un certo colore di sfondo:

```
\pagecolor{red}\afterpage{\pagecolor{white}}
```

Meglio dare i comandi in un posto 'sicuro', l'ottimale è tra capoversi normali, cioè non fra ambienti che possano aggiungere spaziature verticali.

```
\colorbox{<colore>}{<testo>}
\colorbox[<spazio>]{<specifica>}{<testo>}
```

Il primo comando, `\colorbox`, è l'analogo di `\mbox` ma produce una 'box' con colore di sfondo `<colore>`. Vale il discorso di prima per l'argomento opzionale.

```
\fcolorbox{<nome1>}{<nome2>}{<testo>}
\fcolorbox[<spazio>]{<specifica1>}{<specifica2>}{<testo>}
```

È l'analogo di `\fbox`; nella forma senza l'argomento opzionale disegna una cornice di colore `<nome1>` attorno alla 'box' con sfondo di colore `<nome2>`. Nel caso dell'argomento opzionale, i colori vanno dati secondo le regole dello `<spazio>` prescelto: non è possibile usare spazi diversi per definire i due colori. I parametri usati per lo spessore della cornice e per la distanza della cornice dal `<testo>` sono rispettivamente `\fboxrule` e `\fboxsep` (cioè gli stessi usati da `\fbox` e `\framebox`). Anche `\colorbox` usa `\fboxsep` per determinare l'area colorata attorno al `<testo>`.

Va rilevato che né `\fcolorbox` né `\colorbox` permettono sintassi simile a `\framebox` o `\makebox`. Tuttavia non è vietato annidare questi comandi:

```
\fcolorbox{red}{yellow}{\makebox[2cm][c]{Ciao ciao}}
```

produce quasi quanto si vorrebbe senza complicare la sintassi, cioè Ciao ciao. Occorre solo tenere presente che la 'box' non sarà larga esattamente 2 cm, perché si deve aggiungere lo spessore della cornice e lo spazio attorno. Con il pacchetto `calc` potremmo scrivere

```
\fcolorbox{red}{yellow}{%
\makebox[2cm-2\fbboxsep-2\fbboxrule][c]{Ciao ciao}}
```

## 27.6 — Tabella dei colori dello spazio named

GreenYellow		Rhodamine		SkyBlue	
Yellow		Mulberry		Turquoise	
Goldenrod		RedViolet		TealBlue	
Dandelion		Fuchsia		Aquamarine	
Apricot		Lavender		BlueGreen	
Peach		Thistle		Emerald	
Melon		Orchid		JungleGreen	
YellowOrange		DarkOrchid		SeaGreen	
Orange		Purple		Green	
BurntOrange		Plum		ForestGreen	
Bittersweet		Violet		PineGreen	
RedOrange		RoyalPurple		LimeGreen	
Mahogany		BlueViolet		YellowGreen	
Maroon		Periwinkle		SpringGreen	
BrickRed		CadetBlue		OliveGreen	
Red		CornflowerBlue		RawSienna	
OrangeRed		MidnightBlue		Sepia	
RubineRed		NavyBlue		Brown	
WildStrawberry		RoyalBlue		Tan	
Salmon		Blue		Gray	
CarnationPink		Cerulean		Black	
Magenta		Cyan		White	
VioletRed		ProcessBlue			

## 27.7 — Limitazioni

In certi casi particolari l'uso del colore potrebbe cambiare le spaziature del documento rispetto a quando lo si compila senza colore (non con l'opzione `monochrome`, ma proprio senza usare il pacchetto). Per esempio, dare `\pagecolor{<colore>}` fra un ambiente `itemize` e un ambiente `center` può portare a sommare le spaziature che seguono il primo e precedono il secondo, invece che usare solo la massima delle due, come accade normalmente. Analogamente, è meglio non usare `\color` come primo comando in una `minipage` (o `\parbox`) con argomento opzionale `t`, oppure usarlo in tutte quelle da allineare, eventualmente specificando `\color{black}`.

Alcuni driver potrebbero trovarsi in difficoltà se si cerca di colorare il testo o lo sfondo in un ambiente galleggiante oppure si annidano le chiamate di colori. Il driver `pdftex` non dovrebbe soffrire di questo problema.

## 27.8 — Altri pacchetti

Un pacchetto permette di caricare i nomi di molti colori PANTONE<sup>®</sup>. Con

```
\usepackage[pantone]{spotcolor}
```

avremo a disposizione uno spazio di colore `spotcolor` nel quale potremo dare definizioni come

```
\definecolor{miocolore}{spotcolor}{PANTONE129PC,1}
```

La specifica è un nome PANTONE<sup>®</sup> seguito da un numero compreso fra 0 e 1.

Per applicazioni più complesse del colore si consiglia, naturalmente, `xcolor`.

## fancyhdr

`\usepackage{fancyhdr}`

Nei documenti di una certa consistenza è bene fornire al lettore dettagli sul punto in cui si trova; questi dettagli sono molto utili nella consultazione del documento e normalmente si trovano nelle testatine o, più raramente, al piede della pagina. Al piede è normale trovare il numero della pagina, che però può essere anche trasferito nella testatina.

### 28.1 — Comandi

Il modo migliore di chiamare il pacchetto è di scrivere

```
\usepackage{fancyhdr}
\pagestyle{fancy}
\renewcommand{\headrulewidth}{0pt}
```

per poi passare alla personalizzazione vera e propria delle testatine e del piede.

Per capire il modo di funzionamento di `fancyhdr` occorre dire che si immagina la testatina divisa in tre campi: sinistra, centro e destra. Analogamente per il piè di pagina. Se si sta stampando in bianca e volta (fronte-retro), ciascuno dei campi può avere contenuto diverso nelle pagine sinistre e destre. Se si usa l'opzione `oneside`, tutte le pagine sono considerate destre.

Il testo contenuto nei campi riceve il trattamento che ci si attende: si può pensare che il campo di sinistra sia contenuto in un ambiente `flushleft`, quello centrale in un ambiente `center` e quello di destra in `flushright`.

Il comando `\pagestyle{fancy}` inserisce nei sei campi ciò che sarebbe inserito dalla classe in uso, quindi non si ottiene nulla di diverso, almeno per le classi `book` e `report`.

`\fancyhf{}`

Con questo comando vengono azzerati tutti i campi e si è pronti per la personalizzazione delle proprie testatine. Non si dimentichi l'argomento vuoto.

`\fancyhead[⟨campi⟩]{⟨testo⟩}`

Si impostano i campi specificati nell'argomento opzionale; se questo non è specificato, tutti i campi avranno lo stesso contenuto. Ogni campo è indicato con una combinazione di una lettera tra L, C e R (sinistra, centro e destra; *left*, *center* e *right*) e, facoltativamente, da una fra O e E (dispari e pari; *odd* e *even*). Così, per esempio, L0 indica il campo di sinistra nelle pagine dispari, CE il campo centrale nelle pagine pari. Le specificazioni possono essere più di una: con

```
\fancyheader [LO,RE]{...}
```

si specifica che la stessa cosa va nel campo di sinistra delle pagine dispari e in quello di destra delle pagine pari.

Il `⟨testo⟩` è codice L<sup>A</sup>T<sub>E</sub>X che specifica il contenuto del campo, che può quindi variare da pagina a pagina.

`\fancyfoot[⟨campi⟩]{⟨testo⟩}`

Valgono le stesse cose dette per `\fancyhead`, ma ovviamente riguarderanno il piede. Per esempio con

```
\fancyfoot [LE,R0]{\thepage}
```



si specifica che il piede deve contenere il numero di pagina allineato al margine esterno: il sinistro nelle pagine pari e il destro nelle pagine dispari.

`\nouppercase{testo}`

Le classi standard di L<sup>A</sup>T<sub>E</sub>X rendono maiuscolo il contenuto delle testatine. Dando il proprio codice come argomento di `\nouppercase`, questo comportamento viene neutralizzato.

`\leftmark`

Questo comando si espande a ciò che L<sup>A</sup>T<sub>E</sub>X metterebbe normalmente nella testatina delle pagine sinistre, cioè il titolo del capitolo corrente. Vedremo più avanti come modificarne il comportamento.

`\rightmark`

Questo comando si espande a ciò che L<sup>A</sup>T<sub>E</sub>X metterebbe normalmente nella testatina delle pagine destre, cioè il titolo della sezione corrente. Vedremo più avanti come modificarne il comportamento.

`\chaptermark`

Non è un comando di fancyhdr, ma del nucleo di L<sup>A</sup>T<sub>E</sub>X; è modificandone la definizione che si può personalizzare il risultato di `\leftmark`. La definizione usuale è più o meno equivalente a

```
\newcommand\chaptermark[1]{%
  \markboth{\MakeUppercase{\chaptername\ \thechapter. #1}}{}}
```

Qui `#1` è sempre il titolo del capitolo corrente, perché il comando è usato da `\chapter`.

`\sectionmark`

Non è un comando di fancyhdr, ma del nucleo di L<sup>A</sup>T<sub>E</sub>X; è modificandone la definizione che si può personalizzare il risultato di `\rightmark`. La definizione usuale è più o meno equivalente a

```
\newcommand\chaptermark[1]{%
  \markright{\MakeUppercase{\thesection. #1}}}
```

Qui `#1` è sempre il titolo della sezione corrente, perché il comando è usato da `\section`.

`\headrulewidth`

Questo comando contiene lo spessore del filetto di separazione tra la testatina e il corpo del testo. Il suo valore normale è 0,4 pt; va cambiato con `\renewcommand`, come abbiamo mostrato nell'esempio iniziale su come chiamare il pacchetto fancyhdr. Non si usino altri valori che 0 pt e 0,4 pt.

`\footrulewidth`

Questo comando contiene lo spessore del filetto di separazione tra il corpo del testo e il piede. Il suo valore normale è zero e va cambiato con `\renewcommand`. Si noti che i filetti in testa e al piede ricordano molto gli annunci funebri, si eviti di modificare `\footrulewidth`.

`\headheight`

È un parametro dimensionale modificabile con `\setlength` che contiene l'altezza riservata per la testatina. Il pacchetto fancyhdr emette messaggi di avviso suggerendo una dimensione appropriata se quella corrente è troppo piccola rispetto al testo contenuto nella testatina. Meglio aderire alla richiesta, per non avere sorprese durante l'impaginazione.

```
\iffloatpage{<testo1>}{<testo2>}
```

Si può usare `\iffloatpage` in ogni campo o anche nella definizione di `\headrulewidth`. Il `<testo1>` viene usato nelle pagine che contengono solo oggetti galleggianti (figure o tabelle), il `<testo2>` invece viene usato in tutte le altre pagine.

```
\fancypagestyle{<nome>}{<codice>}
```

Si può definire un nuovo stile di pagina (o ridefinirne uno già esistente) avendo a disposizione tutte le funzionalità del pacchetto. L'esempio più tipico è di ridefinire lo stile di pagina `plain`; supponiamo che le nostre testatine siano in corpo ridotto e vogliamo dunque che anche il numero di pagina lo sia, comprese le pagine di inizio capitolo, dove viene usato lo stile `plain`.

```
\fancypagestyle{plain}{%
  % \renewcommand{\headrulewidth}{0pt}%
  \fancyhf{}%
  \fancyfoot[C]{\small\thepage}}
```

Non è necessario ridefinire `\headrulewidth` se l'abbiamo già impostata di default a zero.

## 28.2 — Esempi

Le testatine di questo documento possono essere ottenute con

```
\pagestyle{fancy}
\renewcommand{\headrulewidth}{0.4pt}
\renewcommand{\chaptermark}[1]{%
  \markboth{\ifnum\value{chapter}>0 \thechapter. \fi#1}
  {\ifnum\value{chapter}>0 \thechapter. \fi#1}}
\fancyhf{}
\fancyheader[C]{\nouppercase{\leftmark}}
\fancyfoot[C]{\thepage}
```

Dobbiamo fare in modo che nelle pagine iniziali, dove i capitoli non sono numerati, non esca il numero del capitolo che sarebbe zero. Inoltre vogliamo la stessa testatina nelle pagine destre e sinistre, perciò usiamo anche il secondo argomento di `\markboth` che imposta il `\rightmark`.

Una scelta abbastanza comune per le testatine di un documento nella classe `book` è di avere il titolo del capitolo nelle pagine sinistre e quello della sezione corrente nelle pagine destre, vicini al margine interno. Ai margini esterni della testatina si pone il numero di pagina. Per complicare un po' la cosa comporremo le testatine in neretto, ma in corpo ridotto perché siano evidenti ma non pesanti.

```
\usepackage{fancyhdr}
\pagestyle{fancy}
\renewcommand{\headrulewidth}{0pt}
\fancyhf{}
\fancyhead[LE,RO]{\small\bfseries\thepage}
\fancyhead[RE]{\small\bfseries\nouppercase{\rightmark}}
\fancyhead[LO]{\small\bfseries\nouppercase{\leftmark}}
\fancypagestyle{plain}{%
  \fancyhf{} \fancyfoot{\small\bfseries\thepage}}
\makeatletter
\renewcommand{\chaptermark}[1]{\markboth{%
  \ifnum\value{secnumdepth}>\m@ne
  \if@mainmatter\@chapapp\ \thechapter. \ \fi\fi #1}{}}
\renewcommand{\sectionmark}[1]{\markright{%
  \ifnum\value{secnumdepth}>\z@ \thesection.\ \fi#1}}
\makeatother
```

Gli strani comandi con @ possono essere evitati definendosi nuovi stili di pagina.

```

\usepackage{fancyhdr}
\pagestyle{fancy}
\renewcommand{\headrulewidth}{0pt}
\renewcommand{\chaptermark}[1]{\markboth{#1}{}}
\renewcommand{\sectionmark}[1]{\markright{%
  \ifnum\value{secnumdepth}>0 \thesection.\ \fi#1}}

\fancypagestyle{fancyfront}{%
  \fancyhf{}
  \fancyhead[LE,RO]{\small\bfseries\thepage}%
  \fancyhead[LO]{\small\bfseries\nouppercase{\rightmark}}%
  \fancyhead[RE]{\small\bfseries\nouppercase{\leftmark}}%
}
\fancypagestyle{fancymain}{%
  \fancyhf{}
  \fancyhead[LE,RO]{\small\bfseries\thepage}%
  \fancyhead[LO]{\small\bfseries\nouppercase{\rightmark}}%
  \fancyhead[RE]{\small\bfseries\nouppercase{%
    \chaptername\ \thechapter.\ \leftmark}}%
}
\fancypagestyle{fancyapp}{%
  \fancyhf{}
  \fancyhead[LE,RO]{\small\bfseries\thepage}%
  \fancyhead[LO]{\small\bfseries\nouppercase{\rightmark}}%
  \fancyhead[RE]{\small\bfseries\nouppercase{%
    \appendixname\ \thechapter.\ \leftmark}}%
}
\fancypagestyle{plain}{%
  \fancyhf{}
  \fancyfoot[C]{\small\bfseries\thepage}}

\addto{\frontmatter}{\pagestyle{fancyfront}}
\addto{\mainmatter}{\pagestyle{fancymain}}
\addto{\appendix}{\pagestyle{fancyapp}}
\addto{\backmatter}{\pagestyle{fancyfront}}

```

Per il comando `\addto` è necessario aver caricato `babel`. Questa serie di stili di pagina è abbastanza generica e con le opportune modifiche può soddisfare quasi ogni esigenza. Si ricordi di reimpostare `\headrulewidth` in ogni stile se si vuole il filetto di separazione.

**geometry**

```
\usepackage[\langle opzione \rangle,...] {geometry}
```

Uno dei tormentoni è certamente la modifica della gabbia del documento. Il potente pacchetto `geometry` permette di farlo in modo abbastanza indolore, ma occorre osservare che il gran numero di parametri modificabili rischia di provocare capogiri e risultati poco gradevoli.

Il pacchetto mette a disposizione il comando `\geometry` che evita di dover specificare molte opzioni nella chiamata del pacchetto. Così

```
\usepackage[a4paper,margin=2cm,headheight=14pt]{geometry}
```

e la chiamata alternativa

```
\usepackage{geometry}
\geometry{a4paper,
margin=2cm,
headheight=14pt}
```

sono del tutto equivalenti. Si possono anche dare più comandi `\geometry`, che sono cumulativi. Se si specifica una delle opzioni più volte, è quella finale che conta.

Se si usa questo pacchetto è bene che sia chiamato all'inizio del preambolo, subito dopo i pacchetti standard come `fontenc`, `inputenc` e `babel`. Sicuramente va chiamato e impostato prima di `fancyhdr`.

Il pacchetto ha i suoi default che sono *diversi* da quelli delle classi standard. Quindi chiamare `geometry` senza opzioni, cioè solo con quelle specificate per la classe (come per esempio `a4paper`), o non chiamarlo produce risultati differenti.

**29.1 — Opzioni**

Descriviamo qui le principali opzioni che possono essere date al pacchetto o nel comando `\geometry`; la lista non è completa, descriveremo solo le opzioni più comunemente impiegate. Si noti che, se si è caricato il pacchetto `calc`, si possono sfruttare le sue funzionalità nel passare dimensioni alle varie opzioni.

```
a4paper | letterpaper | b5paper | ...
```

Un formato di pagina; si consulti la documentazione del pacchetto per la lista completa.

```
paperwidth=\langle dimen \rangle
paperheight=\langle dimen \rangle
```

Specificano una larghezza e un'altezza della pagina fisica non standard, per esempio

```
\geometry{paperheight=18.5cm}
\geometry{paperwidth=15cm}
```

```
landscape | portrait
```

Scambia altezza e larghezza della pagina fisica, anche per i calcoli successivi dei parametri di impaginazione.

```
vscale=<numero decimale>  
hscale=<numero decimale>  
scale=<numero decimale>
```

Specifica che l'altezza o la larghezza della gabbia siano la frazione *<numero decimale>* dell'altezza o della larghezza del supporto fisico (la pagina bianca). Per esempio,

```
\geometry{hscale=0.8,vscale=0.7}
```

Con `scale=<numero decimale>` si specifica che `hscale` e `vscale` siano uguali.

```
textwidth=<dimen>  
textheight=<dimen>
```

Specificano la larghezza e l'altezza della gabbia.

```
lines=<numero>
```

Specifica che la gabbia contenga un certo numero di righe di stampa, con l'interlinea valida all'inizio del documento.

```
heightrounded
```

Specifica che l'altezza della gabbia sia arrotondata al numero intero più vicino di righe di stampa.

```
width=<dimen>  
height=<dimen>
```

Sono analoghe a `textwidth` e `textheight`, ma, se sono specificate una o più opzioni di quelle indicate di seguito, la dimensione passata tramite `width` o `height` potrà non coincidere con larghezza o altezza della gabbia.

```
includehead  
includefoot  
includeheadfoot  
includemp  
includeall
```

Con queste opzioni si richiede che si tenga conto della testatina (`includehead`), del piede (`includefoot`), di entrambi (`includeheadfoot`) nella dimensione verticale passata con `height`. Analogamente `includemp` terrà conto dello spazio riservato alle note marginali nella dimensione orizzontale passata con `width`. Impostare `includeall` è equivalente a impostare insieme `includeheadfoot` e `includemp`.

```
ignorehead  
ignorefoot  
ignoreheadfoot  
ignoremp  
ignoreall
```

Analoghe alle precedenti, il significato dovrebbe essere ovvio.

```
left=<dimen> | inner=<dimen>
```

Imposta il margine sinistro, cioè la distanza della gabbia dal margine interno (vicino alla rilegatura). Si può usare un modo o l'altro indipendentemente dal fatto che si stampi solo fronte o fronte e retro.

```
right=<dimen> | outer=<dimen>
```

Imposta il margine destro, cioè la distanza della gabbia dal margine esterno. Si può usare un modo o l'altro indipendentemente dal fatto che si stampi solo fronte o fronte e retro.

```
top=<dimen>
bottom=<dimen>
```

Impostano il margine superiore e inferiore.

```
hmarginratio=<l>:<r>
```

Divide lo spazio orizzontale non occupato dalla gabbia in  $l + r$  parti, assegnandone  $l$  a sinistra (interno) e  $r$  a destra (esterno). I valori di default sono  $l = 1$ ,  $r = 1$  (solo fronte) e  $l = 2$ ,  $r = 3$  (fronte e retro) che sarebbero specificati con

```
\geometry{hmarginratio=1:1}
\geometry{hmarginratio=2:3}
```

```
vmarginratio=<t>:<b>
```

Divide lo spazio orizzontale non occupato dalla gabbia in  $t + b$  parti, assegnandone  $t$  in alto e  $b$  in basso. I valori di default sono  $t = 2$ ,  $b = 3$ .

Per esempio si potrebbe stabilire

```
\geometry{hmarginratio=3:5,vmarginratio=1:2}
```

per ottenere una posizione della gabbia asimmetrica rispetto alla pagina fisica. È buona norma che il margine interno sia minore del margine esterno.

```
bindingoffset=<dimen>
```

Esclude dal conteggio della larghezza della pagina fisica un margine pari a  $\langle dimen \rangle$ . Non si esageri:

```
\geometry{bindingcorrection=0.7cm}
```

è già normalmente abbondante.

```
headheight=<dimen>
headsep=<dimen>
footskip=<dimen>
footnotesep=<dimen>
columnsep=<dimen>
marginparsep=<dimen>
marginparwidth=<dimen>
```

Specificano rispettivamente l'altezza della testatina; la distanza della base della testatina dalla gabbia; la distanza della gabbia dal livello superiore del piede; la distanza delle note al piede dal testo; la separazione tra colonne se è specificata l'opzione di classe `twocolumn`; la separazione tra la gabbia e le note a margine; l'ampiezza riservata per le note a margine.

```
nohead
nofoot
nomarginpar
```

Azzerano le dimensioni riservate per la testatina, il piede o le note a margine, rispettivamente.

## 29.2 — Esempi

Non occorre specificare tutti i parametri: `geometry` è in grado di eseguire i calcoli necessari tenendo conto dei valori di default del pacchetto o della classe.

Questo documento è stato composto con

```
\usepackage[textwidth=388.45pt,textheight=658pt,heightrounded,
headsep=12pt,vmarginratio=1:1]{geometry}
```

Non occorre specificare `hmarginratio` perché la classe è `report`: il documento è pensato per una lettura da schermo e quindi i margini asimmetrici non sono indicati.

Una specifica potrebbe essere la seguente:

- formato  $17 \times 24$ ,
- margine superiore = 2,5 cm,
- margine inferiore = 3,6 cm,
- margine interno = 2,5 cm,
- margine esterno = 3,5 cm,
- margine intestazione = 1,3 cm,
- margine piè di pagina = 2,4 cm.

La sua realizzazione è

```
\geometry{paperheight=24cm,paperwidth=17cm,
  top=2.5cm,bottom=3.6cm,
  headheight=10pt,headsep=\dimexpr 1.3cm-10pt\relax,
  footskip=1.2cm,heightrounded,
  left=2.5cm,right=3.5cm}
```

La giustezza (o larghezza della riga di stampa) e l'altezza della gabbia sono automaticamente impostate. In questo caso si è supposto che la testatina sia composta a corpo 10 e quindi un'altezza di 10 pt sia sufficiente. Si noti l'uso di `\dimexpr` per eseguire i calcoli necessari.

### 29.3 — `\newgeometry`

È possibile cambiare temporaneamente (o anche definitivamente) i parametri di impaginazione dando `\newgeometry{<opzioni>}`. Il comando esegue da sé un `\clearpage` e imposta la nuova pagina secondo le opzioni date come argomento, del tutto simili a quelle descritte in precedenza. Non si possono però usare le opzioni che fissano le dimensioni della pagina.

Per tornare alla precedente impaginazione si dà il comando `\restoregeometry`. Si noti che quando si usa `fancyhdr` e si modifica la larghezza della gabbia occorre anche impostare `\headwidth` alla nuova ampiezza. Si può allora definire un ambiente opportuno

```
\newenvironment{changegeometry}[1]
  {\newgeometry{#1}\setlength{\headwidth}{\textwidth}}
  {\clearpage\aftergroup\restoregeometry}
```

in modo che l'impostazione di `\headwidth` valga solo all'interno dell'ambiente. I parametri di impaginazione vengono dati come argomento di `\begin{changegeometry}`.

```
emptypage
\usepackage{emptypage}
```

Questo pacchetto non fornisce alcun comando. Quando lo si carica si ottiene l'effetto di rendere completamente bianche le pagine che non contengono testo e servono solo per cominciare un capitolo o una parte su una pagina destra. Naturalmente non ha alcun effetto con l'opzione `oneside`.

---

Quest'opera è stata rilasciata sotto la licenza Creative Commons Attribuzione 2.5 Italia. Per leggere una copia della licenza visita il sito web

<http://creativecommons.org/licenses/by-nc-sa/2.5/it/>

o spedisci una lettera a Creative Commons, 559 Nathan Abbott Way, Stanford, California 94305, USA.