Le graffe: queste sconosciute

Claudio Beccari

Sommario

Le parentesi graffe, quadre e tonde sono usate nella sintassi dei comandi LaTeX molto frequentemente, le graffe molto più delle tonde. Tuttavia mentre le parentesi quadre e quelle tonde hanno degli usi molto ristretti, quelle graffe hanno almeno tre interpretazioni diverse e dispongono anche di nomi alternativi. Talvolta si confondono i significati delle graffe o non è chiarissima la loro funzione. Questo tutorial vorrebbe fare chiarezza in questo campo.

Abstract

Braces, brackets and common (round) parentheses play an important rôle in LaTeX syntax; braces in particular are used more often than the other kinds. Nevertheless while brackets and round parentheses play restricted rôles, braces have at least three different interpretations and even some alias names are given to them. Sometimes their rôles are misinterpreted by the user or sometimes their rôles are not so evident. This tutorial is intended to clarify the braces' rôles.

1 Introduzione

Chiunque legga il file sorgente di un qualunque documento LaTeX ne trova alcune parti fitte di parentesi, quasi sempre graffe, talvolta quadre; all'interno dell'ambiente picture si incontrano anche le parentesi tonde. Queste ultime vengono usate solo per delimitare le coordinate dei punti o i coefficienti angolari delle linee oblique nella sintassi di quell'ambiente. Praticamente è difficile trovare altri punti di LaTeX o pacchetti di estensione nei quali si siano usate le parentesi tonde; ovviamente le si incontra spesso nel testo, ma questa loro funzione non ha nulla a che fare con la sintassi di LaTeX.

Le parentesi quadre si incontrano più sovente delle parentesi tonde; esse, nella sintassi di LaTeX, servono per racchiudere le liste delle opzioni per le classi, per molti pacchetti e per alcuni comandi.

Le funzioni delle parentesi graffe sono generalmente descritte come quelle di delimitazione dei gruppi, ma non è proprio così, è solo una mezza verità.

Si consideri la seguente lista di funzioni:

1. {⟨*testo*⟩}
2. ⟨*macro*⟩{⟨*testo*⟩}
3. ⟨*comando*⟩{⟨*testo*⟩}

In questa lista ⟨*testo*⟩ può consistere di qualunque cosa: testo vero e proprio, sequenze di controllo o comandi generici, scatole o qualunque oggetto che abbia senso nel contesto racchiuso dalle graffe.

Il caso 1 rappresenta un gruppo; eventuali definizioni o assegnazioni eseguite dentro il gruppo in modalità non globale[1] perdono la loro efficacia non appena l'interprete dei comandi (tex, ma più sovente pdftex) ha esaurito di elaborare quanto è contenuto nel gruppo e ne esce.

Il caso 2 è forse il caso più diffuso; serve per passare argomenti *non delimitati*[2] a macro definite sia dall'utente stesso sia nel nucleo di LaTeX o nei vari pacchetti. Come è noto gli argomenti possono essere al massimo nove. Gli argomenti non delimitati *devono* essere racchiusi fra graffe se sono composti da più *token* (oggetti), tipicamente da un testo vero e proprio composto da più lettere. Se l'argomento è composto da un solo token le graffe possono essere omesse. In questo senso le graffe sembrano svolgere anche l'azione di raggruppamento visto nel caso 1, ma non è così: quanto viene passato come argomento ad una macro viene sviluppato nel momento in cui la macro viene espansa e se l'argomento conteneva delle assegnazioni o delle definizioni queste possono essere "globali" anche se non sono esplicitamente definite come tali. In ogni caso bisogna rendersi conto che raggruppare dei token per passarli come un unico argomento ad una macro non implica affatto le funzioni del *gruppo* come definito al punto 1.

Il caso 3 con la sintassi LaTeX si presenta assai raramente in modo esplicito, ma è lì in "agguato" pronto per essere utilissimo o, al contrario, per porre in difficoltà l'utente LaTeX che cerca di definire delle macro, ma si trova impacciato da un comportamento che non si aspetta. In realtà questa sintassi è esplicita per molti comandi TeX "primitivi", quelli cioè che non sono definiti come macro, ma appartengono a quei circa 300 comandi del nucleo dell'interprete. Sono quelli in termini dei quali direttamente o indirettamente sono definite tutte le macro.

Questo ultimo caso 3 accetta che le graffe siano sostituite da sinonimi, mentre negli altri casi i

1. Per il significato di comando globale, di assegnazione globale, di definizione globale si vede il manuale di Knuth (1996). Alternativamente si veda la documentazione di Gregorio (2009).
2. Per i comandi delimitati, oltre che nel testo di Knuth (1996), si può anche vederne una applicazione nell'articolo di Beccari (2008b).

sinonimi non sono accettabili o lo sono in certe circostanze come si vedrà più avanti; spesso si ha la sensazione che i comandi primitivi vogliano gli oggetti su cui operare raccolti come nel caso 2 o formino un gruppo come nel caso 1, ma in realtà non è così sia perché loro o i loro sinonimi sono sempre obbligatori, anche nel caso di un solo oggetto, sia perché la funzione di limitare la validità delle definizioni o delle assegnazioni può essere assente.

2 I sinonimi delle graffe

Per vari motivi le graffe possono essere indicate con i loro sinonimi; il nucleo di LaTeX esegue le associazioni:

```
\let\bgroup {
\let\egroup }
```

Il comando `\let` è un comando primitivo che serve per associare un token ad un altro token[3]; questi token associati prendono anche il nome di *token impliciti* ma la terminologia non è così importante; il fatto che il token associato, detto gergalmente *alias* del token vero, ne conserva tutte le caratteristiche e spesso può essere usato in alternativa al token vero. Non è sempre vero, però: quando l'interprete legge dal file di ingresso i singoli caratteri che lo costituiscono, esegue la *tokenizzazione*; generalmente ogni segno viene definito come un token che rappresenta il segno stesso corredato da un codice di categoria; così avviene per le lettere dell'alfabeto, per le cifre, per i segni di interpunzione, per le varie parentesi, per lo spazio, per i caratteri proibiti, per i caratteri che l'interprete sa essere attivi perché dichiarati tali prima di leggere il file sorgente, eccetera. Quando l'interprete incontra una *contol sequence*, una fila di lettere precedute dal backslash o un solo carattere diverso da una lettera preceduto dal backslash, allora forma un solo token formato dal nome della control sequence a cui associa il codice di categoria specifico dei comandi o delle macro.

In questo modo `\bgroup` e `\egroup` hanno lo stesso codice e la stessa categoria rispettivamente della graffa aperta e di quella chiusa, ma restano ancora un pochino diversi. Infatti quando l'interprete legge il file di ingresso esso tiene conto delle graffe aperte e di quelle chiuse in modo da assicurare che esse siano correttamente bilanciate. In questo caso `\bgroup` e `\egroup` non partecipano al conteggio delle graffe aperte e chiuse e quindi possono apparire anche scompagnate all'interno di un altra coppia di graffe esplicite correttamente accoppiate. Vedremo più avanti quanto sia utile questa caratteristica.

3. Per il significato di token, oltre al testo di Knuth (1996) si può vere anche l'articolo di Beccari (2008a).

3 I gruppi

Una espressione racchiusa fra graffe, senza che queste servano per passare dei token raggruppati a una macro o a un comando primitivo, come si è detto, forma un gruppo. I gruppi possono anche essere racchiusi fra

`\begingroup` ... `\endgroup`

oppure fra

`\bgroup` ... `\egroup`

ma questa seconda scrittura appare barocca, lunga da scrivere e, se non ci sono i motivi particolari che vedremo più avanti, sarebbe meglio evitarla.

Dentro il gruppo, senza che l'utente debba preoccuparsi di questi dettagli, viene costituito uno *stack* nel quale vengono memorizzati tutti i valori dei registri, delle *control sequence* e dei caratteri attivi esistenti via via che si attribuisce loro un nuovo significato; chiudendo il gruppo, lo stack viene usato per ripristinare i valori precedenti alle "variabili" che sono state modificate dentro il gruppo. Questo avviene sempre quando le ridefinizioni o le assegnazioni non sono globali; se una assegnazione o una definizione è globale, il valore precedente della variabile viene perso perché, appunto, l'operazione è globale e nulla viene immesso nello stack. L'uso dei gruppi serve proprio per ottenere questa funzionalità.

Merita segnalare che la coppia `\begingroup` e `\endgroup` mantiene l'interprete nella stessa modalità operativa, modo testo, modo matematico, modo verticale, modo orizzontale, eccetera. Invece le coppie di graffe esplicite o implicite possono, non necessariamente devono, anche prevedere un cambiamento di modo. In generale gli ambienti di LaTeX ricorrono a `\begingroup` incluso nella definizione di `\begin` e `\endgroup` incluso nella definizione di `\end`.

Eseguendo qualunque tipo di definizione mediante uno dei comandi LaTeX oppure TeX, quali `\newcommand`, `\providecommand`, `\renewcommand`, `\def`, `\edef`, la definizione resta locale al gruppo; vale a dire che quando si esce dal gruppo la definizione si perde.

Quando si esegue qualche assegnazione non globale mediante comandi TeX, quali ad esempio `\count...=`, `\advance`, `\multiply`, `\divide`,... (e analogamente per le assegnazioni a tutti gli altri registri di TeX e LaTeX, come le lunghezze, le scatole, eccetera), questa assegnazione resta in vigore solo dentro il gruppo ma, uscendo dal gruppo, ogni registro viene ristabilito allo stato che esso aveva prima di entrare nel gruppo.

Quando si devono eseguire assegnazioni o definizioni complesse usando registri interni o macro interne di TeX o LaTeX, è conveniente eseguire tutte le manipolazioni all'interno di un gruppo, proprio per ristabilire tutto, ogni registro e ogni

macro interna, allo *statu quo ante* cosicché nulla sia alterato nella memoria dell'interprete.

Dentro il gruppo possono venire "scritte" del cose che vanno a finire nella scatola 255, quella dove si accumula tutto quanto è stato composto fino a quel momento e dal quale ogni tanto la routine di output estrae l'equivalente di una pagina da accodare al file di uscita. In questo caso il gruppo serve per riportare tutto allo statu quo ante, tranne che per lo stato della scatola 255 che per la sua funzione è sempre un po' speciale e assolutamente intoccabile direttamente dall'utente.

Può succedere però che dentro il gruppo si eseguano delle elaborazioni il cui risultato finale debba essere usato fuori dal gruppo. Si hanno due possibilità:

1. L'assegnazione o la definizione del risultato finale va eseguito con i comandi TEX che a loro volta vanno preceduti dal comando primitivo `\global`; oppure
2. l'assegnazione o la definizione va eseguita ricorrendo al procedimento che vedremo fra poco.

Nel primo caso l'assegnazione o la definizione diventano globali in assoluto e restano valide anche quando l'interprete sia uscito da ogni qualsivoglia gruppo nel quale si trovava al momento della definizione o assegnazione. Questo fatto potrebbe non essere quello che si desidera; in questo caso bisogna ricorrere al secondo procedimento che fa uso degli alias.

Mi spiego con un esempio; supponiamo di voler calcolare quanto sia lungo l'alfabeto minuscolo latino con un determinato font, che supporremo essere quello in vigore al momento del comando che esegue questa determinazione (ovviamente facciamo finta non non sapere che esiste il comando LATEX `\settowidth`); definiremo allora:

```
\newlength{\alfabeto}

\newcommand*\lunghezzalfabeto{%
\bgroup \setbox0
  \hbox{abcdefghijklmnopqrstuvwxyz}%
\edef\x{\noexpand\egroup
  \noexpand\alfabeto=\the\wd0}
\x}
```

in modo che dopo aver dato il comando `\lunghezzalfabeto` possiamo sapere che l'alfabeto corrente minuscolo è lungo `127.58365pt`.

La definizione della macro in questione comincia con l'apertura di un gruppo mediante l'alias della graffa aperta; imposta poi la scatola numero 'zero' con l'alfabeto minuscolo corrente; poi, e qui comincia il bello, esegue una definizione espansa del comando `\x` dove in realtà espande solo il comando `\the`, che è quello che serve per scrivere il contenuto di un registro o di un comando relativo a un registro; in questo caso si tratta dell'espressione `\wd0` che indica la larghezza (width) della scatola 'zero'; questo valore, proprio perché si tratta di una definizione espansa non viene ancora assegnato al registro dimensionale `\alfabeto` (definito poco sopra) ma viene semplicemente messo nel testo della definizione; i comandi primitivi `\noexpand` servono per evitare che durante l'espansione vengano espansi prematuramente i comandi `\egroup` e l'equivalente in termini di registri dimensionali corrispondente ad `\alfabeto`. Alla fine dell'espansione è come se la macro `\x` fosse stata definita con

```
\def\x{\egroup\alfabeto=127.58365pt}
```

Completata questa definizione la macro `\x` viene eseguita; questo significa che nel buffer d'entrata dell'interprete il nome della macro viene sostituito con la sua definizione e questa a sua volta viene eseguita; per prima cosa si chiude il gruppo (e con ciò si restituisce allo statu quo ante sia lo stato della scatola 'zero' – usata molto spesso nel nucleo di LATEX per gli scopi più diversi – sia il significato della macro `\x` stessa) poi, ormai fuori dal gruppo, il valore della lunghezza dell'alfabeto viene effettivamente assegnato al registro dimensionale `\alfabeto`.

Questo espediente garantisce che il valore calcolato possa essere "lanciato" oltre la chiusura del gruppo, senza lasciare tracce dietro di sé. È superfluo evidenziare che questo espediente è usato spesso all'interno del nucleo di LATEX e di numerosissimi pacchetti di estensione.

4 Gli argomenti delle macro

Ogni argomento obbligatorio non delimitato formato da più di un token deve venire passato alla macro racchiuso fra parentesi graffe esplicite. In questo caso non si possono assolutamente usare le graffe implicite mediante i comandi `\bgroup` e `\egroup` perché l'interprete riconosce un argomento di macro proprio riconoscendo le graffe esplicite e contando quante se ne aprono e quante se ne chiudono perché l'argomento composto di molti token, che può comprendere altre possibili graffe adeguatamente appaiate, termina quando l'interprete arriva a riconoscere la parentesi graffa chiusa che si appaia alla prima graffa aperta dopo il nome della macro.

Questo implica un fatto importante: se si crea una macro nella cui definizione compare un'altra macro che agisce sui suoi argomenti, questa deve avere la graffa esplicita aperta e quella chiusa entrambe contenute nella definizione della macro che la contiene. Sembra del tutto ovvio, diamine!

Al momento opportuno si cade in questo trabocchetto e lo dimostro con un esempio: vogliamo creare un ambiente che permetta di incorniciare un testo qualunque. Sappiamo che `\framebox` può

incorniciare un testo che si sviluppa su una sola riga; per incorniciare un testo che si svolge su più righe bisogna passare attraverso un ambiente intermedio come minipage; proviamo un po' e scopriamo che il seguente codice, già abbastanza elaborato, funziona:

```
\framebox{\textwidth}{%
\begin{minipage}{%
\dimexpr\textwidth-2\fboxsep
-2\fboxrule}
Testo di qualunque lunghezza che comporti
un numero di righe maggiore di uno.
\end{minipage}}
```

Bene: allora possiamo creare una macro che possa ricevere come parametri la specificazione della larghezza e il testo; la specificazione della larghezza sia facoltativa, ma il valore di default sia la larghezza della riga corrente:

```
\newcommand\cornice[2][\linewidth]
\framebox[#1]{%
\begin{minipage}{%
    \dimexpr#1-2\fboxsep-2\fboxrule}
#2\end{minipage}}
```

Collaudiamo il nuovo comando e constatiamo funziona con il testo di prova: "Testo di qualunque lunghezza che comporti un numero di righe maggiore di uno."

Benissimo, siamo a cavallo; allora possiamo creare un ambiente:

```
\newenvironment{cornice}[1][\linewidth]{%
%                                   apertura
\framebox{\begin{minipage}{%
    \dimexpr#1-2\fboxsep-2\fboxrule}}
{%                                  chiusura
\end{minipage}}}
```

Verifichiamo l'ambiente e riceviamo subito un messaggio d'errore al quale non sappiamo rispondere. Che cosa è successo? Semplicemente che le graffe che racchiudono l'argomento di `\framebox` sono in comandi diversi; la graffa aperta si trova dentro la definizione del comando di apertura dell'ambiente, mentre la graffa chiusa si trova dentro la definizione del comando di chiusura dell'ambiente. Ciò non è possibile nel modo più assoluto e non c'è verso di risolvere il problema con gli alias delle graffe.

Non solo ma ripiegando sul comando `\cornice`, definito sopra e che aveva passato il collaudo, scopriamo che se il testo non è di un solo capoverso, il comando cessa di funzionare. Diamine, che cosa non va? Lo vedremo nel prossimo paragrafo.

5 I delimitatori degli oggetti

I comandi primitivi di TEX spesso ricevono i loro argomenti o gli oggetti su cui devono operare senza fare ricorso alle graffe. I comandi di definizione usano le graffe per racchiudere la definizione di nuove macro e di nuovo queste graffe devono essere esplicite come per le macro definite. Ma usano le graffe anche i comandi per usare le scatole. A questi ci riferiremo in particolare, perché le loro graffe possono essere anche implicite.

LATEX consente all'utente solo di usare scatole orizzontali, dentro le quali esso opera in "LR mode", cioè scrivendo solo in orizzontale, tipicamente da sinistra (L = left) a destra (R = right); in realtà anche LATEX apparentemente offre altre scatole, oltre a quelle definibili con `\newsavebox` e usabili con `\sbox`, `\savebox`, `\usebox`: `\mbox`, `\makebox`, `\fbox`, `\framebox`. In realtà si tratta quasi solamente di comandi di gestione di scatole che in generale vengono riempite, talvolta incorniciate, ma spesso usate immediatamente senza conservarle in memoria. Tutte però sono sostanzialmente scatole orizzontali, anche se talvolta si riesce a caricarle con scatole verticali non accessibili direttamente.

La loro limitazione è che comunque il loro argomento non può essere altro che una linea di testo o anche un capoverso già composto ma ben nascosto dentro una scatola verticale usata tramite il comando `\parbox` o l'ambiente minipage. L'argomento di quei comandi non deve avere nessun comando che possa indurre la macro a credere che il suo argomento sia costituito da più di un capoverso. Ecco come si dà una risposta all'interrogativo posto alla fine del paragrafo precedente.

TEX offre all'utente una varietà di scatole e di comandi per gestirle. Le scatole sono sostanzialmente `\hbox` (che coincide con il tipo di scatola disponibile agli utenti di LATEX), `\vbox`, `\vtop` e `\vcenter`, più una notevole varietà di comandi per gestire, copiare, svuotare quelle scatole. Tutti questi comandi primitivi di caricamento delle scatole richiedono che gli oggetti da caricare siano racchiusi fra graffe esplicite o implicite. In altre parole se l'oggetto da inserire in una scatola fosse fatto da un solo token, non sarebbe possibile omettere le graffe esplicite o implicite che siano; le graffe sono funzionali a questi comandi primitivi e l'interprete le cerca esplicitamente o implicitamente perché in loro assenza esso non sarebbe in grado di usare la scatola.

Le tre scatole verticali disponibili con TEX (e usate internamente da LATEX per costruire il risultato di `\parbox` e dell'ambiente minipage) differiscono per la posizione del punto di riferimento: `\vbox` ha il suo punto di riferimento coincidente con quello dell'ultimo oggetto verticale (in basso) che ha impilato, tipicamente il punto di riferimento dell'ultima riga del suo contenuto; `\vtop`, invece, ha il suo punto di riferimento coincidente con quello del primo oggetto verticale (in alto) che ha impilato, tipicamente il punto di riferimento della prima riga che la scatola contiene. `\vcenter` è un po' speciale perché il suo punto di riferimento, a metà della

sua altezza complessiva, viene collocato sull'asse matematico della riga dove la scatola viene a trovarsi; per cui questa scatola deve venire usata solo in ambiente matematico, anche se il suo contenuto è testuale.

Ecco allora che con le graffe implicite e un po' di programmazione di basso livello possiamo ora risolvere il problema di incorniciare un testo formato di più capoversi. Si esamini il codice:

```
\newenvironment{cornice}[1][\linewidth]{%
\fboxsep=6pt \setbox0\vbox\bgroup
\hsize=\dimexpr#1-2\fboxsep-2\fboxrule
\@parboxrestore
}{\egroup\framebox{\box0}}
```

Ecco subito un esempio dell'uso di questo ambiente:

> Questo è un testo abbastanza lungo per occupare più di due righe in questo articolo composto su due colonne.
> Come si vede, infatti, il secondo capoverso, ancora composto di più righe, si accoda al primo dentro la cornice.

Esso risulta prodotto dal seguente testo nel file sorgente:

```
Ecco subito un esempio dell'uso di questo
ambiente:  \begin{center}
\begin{cornice}[0.9\columnwidth]
Questo è un testo abbastanza lungo per
occupare più di due righe in questo
articolo composto su due colonne.

Come si vede, infatti, il secondo
capoverso, ancora composto di più righe,
si accoda al primo dentro la cornice.
\end{cornice}\end{center}
Esso risulta prodotto dal seguente testo
nel file sorgente:
```

Il commento al codice ora non dovrebbe presentare misteri: siccome stiamo parlando di un ambiente, esso forma implicitamente un gruppo, quindi tutto quello che vi definiamo dentro o le variabili a cui assegniamo dei valori o dei contenuti vengono ripristinate dopo l'uscita dall'ambiente. Pertanto l'assegnazione di un nuovo valore alla variabile dimensionale `\fboxsep` non ha conseguenze sul resto del documento quando l'ambiente verrà chiuso. Si apre poi una scatola verticale del tipo `\vbox` da conservare nella scatola 'zero'; il suo contenuto comincia subito dopo una graffa implicita aperta; siccome dobbiamo comporre il testo in modo verticale (dobbiamo costruire uno o più capoversi) dobbiamo fissare la giustezza della composizione che, a livello delle scatole di TEX, si chiama `\hsize`; in particolare, qualunque larghezza vogliamo che abbia la cornice, quella di default o quella che noi specifichiamo come argomento opzionale all'apertura dell'ambiente, dobbiamo tenere conto che attorno al testo composto apparirà uno spazio attorno ai quattro lati (la cui larghezza è stabilita mediante `\fboxsep`) e, per quanto piccolo, bisogna tenere conto dello spessore del filetto che forma la cornice (il cui valore è contenuto in `\fboxrule`); ecco perché si chiede a TEX di determinare il risultato di una espressione dimensionale con `\dimexpr`, che esegue i calcoli necessari. Fatto questo si completano i comandi di apertura con un comando `\@parboxrestore`, che serve per impostare i parametri compositivi di default all'interno della scatola verticale. Bisogna ora provvedere ai comandi di chiusura: questi cominciano con una graffa chiusa implicita che chiude anche la scatola verticale; il suo contenuto è ora nella scatola 'zero'; finalmente il comando `\framebox` incornicia la scatola 'zero'.

Come ci si rende conto, queste operazioni non sarebbero state possibili se le graffe di apertura e di chiusura delimitanti il contenuto della scatola verticale non fossero state delle parentesi implicite, quella aperta dentro i comandi di apertura e quella chiusa dentro i comandi di chiusura.

6 Conclusioni

Con questo tutorial si è cercato di chiarire la funzione delle parentesi graffe aperte e chiuse a seconda di come vengano usate e quindi a seconda del contesto; si è anche mostrata la differenza fra i comandi di gruppo espliciti e impliciti. È chiaro che si potrebbero fare molti altri esempi; il lettore curioso non ha altro da fare se non aprirsi il suo pacchetto preferito oppure il file `latex.ltx` per trovare altri esempi, lo studio dei quali non può che giovargli se desidera costruirsi delle macro che usino le coppie di graffe nei vari modi descritti in questo articolo.

Riferimenti bibliografici

Beccari, C. (2008a). «I registri token: questi sconosciuti». *ArsTEXnica*.

— (2008b). «Macroistruzioni con argomenti delimitati». *ArsTEXnica*.

Gregorio, E. (2009). «Appunti di programmazione in TEX e LATEX».

Knuth, D. E. (1996). *The TEXbook*. Addison Wesley, Reading, Mass., 16ª edizione.

▷ Claudio Beccari
Villarbasse (TO)
`claudio dot beccari at gmail dot com`