

La composizione di tabelle con larghezza specificata

Claudio Beccari

Sommario

Questo tutorial si propone di esaminare il funzionamento elementare dei comandi del nucleo di \LaTeX che servono per produrre le tabelle con larghezza specificata; ne esamina pregi e difetti e, come esercizio, ne propone di risolvere alcuni difetti con macro apposite.

Abstract

This tutorial examines the \LaTeX kernel basic commands necessary to typeset tabular material with specified width; pros and cons are discussed and, as an exercise, this tutorial suggests to correct some glitches with suitable macros.

1 Introduzione

Gli utenti di \LaTeX usano l'ambiente `tabular` più o meno frequentemente e ne apprezzano la funzionalità anche se spesso non ne sfruttano appieno le potenzialità; per altro il sistema \TeX ormai viene distribuito con numerosi pacchetti di estensione per la composizione delle tabelle, dei quali quello maggiormente usato sembra essere `array`.

Molto più raramente gli utenti compongono tabelle di larghezza prefissata mediante l'ambiente `tabular*`; questo non solo dipende dal fatto che le tabelle di larghezza prefissata sono meno frequenti, ma anche perché i risultati ottenibili mediante l'ambiente `tabular*` talvolta sono ben lontani dall'ottimo.

Si consideri la tabella 1 che contiene una tabella della sua larghezza naturale in base ai parametri compositivi che \LaTeX usa per la composizione delle tabelle.

La tabella in questione contiene anche i filetti verticali; sono stati usati apposta per descrivere il funzionamento dei comandi, anche se non è considerato buono stile quello di usare sia i filetti verticali sia i filetti orizzontali¹. Per altro si vede che dentro ogni colonna i contenuti delle varie celle sono separati dai filetti di un piccolo spazio il cui ammontare *minimo* è specificato dal parametro interno `\tabcolsep` che è predefinito al valore di 6 pt, pari a circa 2 mm. La stessa tabella composta senza questo spazio di separazione apparirebbe come nella tabella 2 e, come si vede, è di una bruttezza inaccettabile.

1. I filetti orizzontali per iniziare e finire la tabella e per separare la righe delle intestazioni dal corpo della tabella sono ammessi.

Se si desiderasse comporre la stessa tabella specificandone la larghezza, questa, ovviamente, non potrebbe essere specificata di valore inferiore a quello della tabella 2, ma, oltre ad usare l'ambiente `tabular*`, bisogna anche provvedere ad un allungamento variabile in ogni cella in modo che \LaTeX possa estendere ogni casella quanto basta perché l'intera tabella abbia la larghezza complessiva specificata. Questo, dice il manuale \LaTeX , LAMPORT (1994), si ottiene mediante una *@-espressione* che specifichi l'ammontare di questo spazio allungabile e di quanto esso possa venire allungato. Il manuale \LaTeX ricorda all'utente anche che questo spazio allungabile viene inserito a sinistra del contenuto di ogni colonna a partire dalla colonna che segue quella che contiene la *@-espressione*. Non ha importanza se la *@-espressione* sia collocata come separatore di sinistra o di destra della cella; l'importante è che la cella delimitata dalla *@-espressione* non viene toccata da questa definizione di spazio aggiuntivo allungabile.

Rivediamo allora che cosa succede se specifichiamo l'allungamento come separatore di sinistra dalla prima colonna di ogni riga, dando come direttiva di apertura della tabella il comando

```
\begin{tabular*}{\textwidth}{%  
  @{\extracolsep{\fill}}\vline}*5{c|}}
```

Il risultato si vede nella tabella 3 e di nuovo è di una bruttezza incredibile; ma che cosa è andato storto? Eppure si sono seguite le istruzioni indicate nel manuale \LaTeX .

2 Gli inconvenienti della composizione delle tabelle

Gli esempi mostrati nell'introduzione usano esclusivamente i comandi del nucleo di \LaTeX ; i risultati, specialmente per la tabella estensibile 3 non sono entusiasmanti, anzi sono pessimi.

Seguendo le raccomandazioni contenute in PANTIERI (2009) e PANTIERI (2008), oltre a quelle riportate nella guida GUIT (2009), si possono trovare molte buone idee e raccomandazioni per ovviare agli inconvenienti visti. Fra tutte, entro certi limiti, è bene tenere conto della possibilità di comporre la tabella secondo le impostazioni di default, senza cercare di estenderla ad una larghezza specificata, e poi ingrandirla o rimpicciolirla mediante il comando `\resizebox` messo a disposizione dal pacchetto `graphicx`; infatti, quando la larghezza naturale con i parametri di default è molto simile a quella desiderata, la soluzione migliore, forse, è

TABELLA 1: Tabella composta con i parametri predefiniti del nucleo di L^AT_EX

Pippo	Pluto	Paperino	Topolino	Minnie
Tizio	Caio	Sempronio	Giulio	Cesare

TABELLA 2: Tabella composta con `\tabcolsep=0pt`

Pippo	Pluto	Paperino	Topolino	Minnie
Tizio	Caio	Sempronio	Giulio	Cesare

TABELLA 3: Tabella composta con la larghezza specificata pari alla giustezza, ma con un errore relativo alla prima colonna

Pippo	Pluto	Paperino	Topolino	Minnie
Tizio	Caio	Sempronio	Giulio	Cesare

TABELLA 4: Tabella composta con la larghezza specificata pari alla giustezza senza l'errore relativo alla prima colonna

Pippo	Pluto	Paperino	Topolino	Minnie
Tizio	Caio	Sempronio	Giulio	Cesare

proprio quella di ingrandire o rimpicciolire la tabella per quella via. Negli esempi usati in questo articolo la larghezza naturale è piuttosto piccola rispetto a quella desiderata, perciò la soluzione suggerita non sarebbe adeguata.

Tuttavia qui ci concentriamo sulle possibilità offerte dai comandi contenuti nel nucleo di L^AT_EX oppure dai comandi più o meno primitivi del linguaggio T_EX.

I brutti risultati visti prima dipendono da due fattori; (a) un errore facile da commettere (e qui espressamente commesso) e (b) una incapacità di L^AT_EX di comporre le tabelle estensibili in modo accettabile. Usando pacchetti esterni, per esempio `tabularx.sty`, si potrebbero forse avere risultati migliori, ma non con una tabella così semplice come quella degli esempi appena illustrati.

L'errore (espressamente) commesso consiste nell'aver "dimenticato" che la *@-espressione* sostituisce completamente il separatore fra le colonne; nel nostro caso essa sostituisce il "separatore" del margine a sinistra della prima colonna; come si vede il contenuto della prima cella della prima colonna non è separato dal filetto verticale alla sua sinistra; ci si è (volutamente) dimenticati di reinserire lo spazio di separazione fra il filetto verticale `\vline`² e il contenuto della prima colonna. La prima riga dell'ambiente `tabular*` avrebbe dovuto essere stata scritta correttamente in questo modo:

```
\begin{tabular*}{\textwidth}{%
  @{\extracolsep{\fill}}\vline
  \hspace{\tabcolsep}*5{c|}}
```

Il risultato 4 continua a non essere accettabile, anche se ora il contenuto della prima colonna è correttamente distanziato dal filetto alla sua sini-

2. All'interno delle *@-espressioni* bisogna usare per il filetto verticale il comando `\vline` invece della più semplice specificazione `|`.

stra. Si vede però che l'allungamento estensibile inserito con la parte di *@-espressione* che recita

```
\extracolsep{\fill}
```

mostra il suo effetto solo a partire dalla seconda colonna e solo a sinistra del contenuto della seconda e delle restanti colonne.

Come provvedere a questa bruttezza?

Per quanto riguarda l'allungamento estensibile, basterebbe inserire delle colonne fittizie in modo che queste colonne contengano solo l'allungamento. Il codice di composizione della tabellina che abbiamo finora usato come esempio:

```
\begin{tabular*}{\textwidth}{%
  @{\extracolsep{\fill}}\vline
  \hspace{\tabcolsep}*5{c|}}
\hline
Pippo & Pluto & Paperino & Topolino
& Minnie \\
Tizio & Caio & Sempronio & Giulio
& Cesare \\
\hline
\end{tabular*}
```

dovrebbe venire modificato così:

```
\setlength{\tabcolsep}{0pt}%
\begin{tabular*}{\textwidth}{%
  @{\extracolsep{\fill}}c|*5{cc|}}
\hline
&Pippo && Pluto && Paperino && Topolino
&& Minnie& \\
&Tizio && Caio && Sempronio && Giulio
&& Cesare& \\
\hline
\end{tabular*}
```

La tabella 5 è ora composta correttamente; i filetti verticali compaiono tutti esattamente equidistanziati dai contenuti delle varie colonne, per cui

TABELLA 5: Tabella estensibile contenente colonne fittizie

Pippo Tizio	Pluto Caio	Paperino Sempronio	Topolino Giulio	Minnie Cesare
----------------	---------------	-----------------------	--------------------	------------------

TABELLA 6: Tabella composta con `\tabcolsep=9.87mm`

Pippo Tizio	Pluto Caio	Paperino Sempronio	Topolino Giulio	Minnie Cesare
----------------	---------------	-----------------------	--------------------	------------------

TABELLA 7: Tabella composta con `widetable`

Pippo Tizio	Pluto Caio	Paperino Sempronio	Topolino Giulio	Minnie Cesare
----------------	---------------	-----------------------	--------------------	------------------

questi contenuti sono tutti correttamente centrati fra i filetti verticali.

Tuttavia il “trucco” di azzerare la spaziatura standard (`\setlength{\tabcolsep}{0pt}`) e di aggiungere le colonne fittizie (portandole ad un totale di 11 colonne, contro le 5 effettivamente necessarie) sembra un artificio veramente acrobatico, anche se in realtà il TEXbook, KNUTH (1996), mostra esempi di tabelle composte con i comandi primitivi del sistema TEX, o di quelli del plain TEX, che attribuiscono deliberatamente una colonna ad ogni separatore; se si usa plain TEX la cosa è accettabile, ma se si usa LATEX bisognerebbe evitare di ricorrere a una programmazione di così basso livello; tuttavia non scordiamoci che questa programmazione è la più efficace in quanto permette una maggiore elasticità.

Per esempio, se si volesse comporre la stessa tabella senza ricorrere ai filetti verticali ma specificando comunque una larghezza pari alla giustezza, sarebbe forse preferibile evitare gli spazi allungabili a sinistra della prima colonna e a destra dell’ultima; in questo caso, avendo specificato una colonna apposta per i separatori non sarebbe difficile modificare il codice presentato prima; il risultato sarebbe proprio quello voluto.

Tuttavia non dobbiamo scordarci che prima di comporre qualunque tabella sarebbe meglio studiarla attentamente per scegliere il tipo di composizione più appropriato, senza pretendere di comporre delle brutte tabelle al fine di rispettare specifiche di progetto tipo: “tutte le tabelle devono avere una larghezza pari alla giustezza”. Non è accettabile un approccio simile, mentre è accettabile “ogni tabella deve avere la larghezza che più le si addice in base al suo contenuto”.

3 Per evitare le acrobazie

Il titolo di questa sezione potrebbe sembrare una presa in giro. Infatti qui ci si propone di indicare come ottenere una tabella di larghezza specificata senza ricorrere alle colonne fittizie. Vedremo che le acrobazie sarà necessario farle in altri modi, quindi,

comunque sia, le tabelle di larghezza specificata richiedono sempre qualche sforzo in più per venire composte in modo decente.

3.1 Prima soluzione manuale

Se riprendiamo la tabella 2, che, conviene ricordarlo, non è composta con l’ambiente `tabular*` ma con il semplice `tabular`, e ne misuriamo la larghezza, possiamo determinare quanta larghezza manca affinché essa sia pari alla larghezza specificata; infatti questa larghezza mancante deve essere distribuita fra i vari separatori `\tabcolsep` che compaiono nella tabella stessa; essa ha 5 colonne, e ognuna ha una spaziatura `\tabcolsep` a sinistra e una a destra; dunque ogni `\tabcolsep` deve essere 1/10 della larghezza mancante. Nel nostro caso la larghezza mancante vale circa 99 mm, per cui ogni `\tabcolsep` deve valere circa 9,9 mm. Il “circa” sta a indicare che con un righello millimetrato è difficile essere più precisi; questo non è un vero problema: basta infatti specificare `\tabcolsep=9.9mm` e poi eventualmente correggere in più o in meno di qualche centesimo di millimetro; dopo due o tre passate di bozze si vede che `\tabcolsep=9.87mm` è perfettamente adeguato e il risultato è mostrato nella tabella 6.

Se si deve comporre una sola tabella con questa “acrobazia” manuale, il lavoro extra da fare non è poi così grande; ed è perfettamente accettabile. Se lo si deve fare per molte tabelle, allora sarebbe meglio disporre di una macro.

3.2 Seconda soluzione automatica

Nessuna macro è onnipotente, ma è possibile eseguire i calcoli con tutta la precisione di cui TEX, il programma, è capace, purché si disponga di una macro adeguata.

A mano si è misurata la larghezza mancante e la si è divisa per il numero di `\tabcolsep` presenti nella tabella. In realtà la tabella potrebbe contenere altre *@-espressioni* e il numero di `\tabcolsep` potrebbe non essere uguale al doppio del numero delle colonne.

Conviene allora far sì che TEX stesso calcoli quanti `\tabcolsep = t` contribuiscano alla larghezza della tabella; con qualche acrobazia si potrebbe ottenere che TEX stesso conti il numero di segni & che compaiono in ogni riga, ma la presenza di comandi `\multicolumn` potrebbe rendere incerto il calcolo.

Per evitare questa incertezza conviene allora misurare le due larghezze l_0 ed l_1 che la tabella assume quando `\tabcolsep` vale zero o, rispettivamente, un valore convenzionale t_1 , per esempio di 6 pt. Un semplice calcolo di proporzioni permette allora di dire: se con $t = 0$ ottengo l_0 e con $t = t_1$ ottengo l_1 quanto deve valere t per ottenere la larghezza specificata l_{spec} ? Un semplice esercizio di proporzioni permette di calcolare che

$$t = t_1 \frac{l_{\text{spec}} - l_0}{l_1 - l_0} \quad (1)$$

Il calcolo è banale, ma la macro deve comporre la tabella tre volte, le prime due delle quali servono solo per determinare l_0 ed l_1 ; dopo queste due prime composizioni deve eseguire due sottrazioni di lunghezze, una divisione fra lunghezze, e una moltiplicazione di una lunghezza per il quoziente precedentemente calcolato. Tutto semplicissimo, salvo il fatto che TEX esegue solo divisioni fra numeri interi e ne può calcolare il rapporto, ma in aritmetica intera, quindi senza determinare la parte fratta del quoziente.

Per quante richieste il sottoscritto abbia fatto al LATEX Team, a tutt'oggi non esiste un algoritmo definito mediante un comando primitivo che consenta a TEX di calcolare un quoziente fratto. Esistono solo delle macro (abbondantemente usate nel nucleo di LATEX, specialmente per le operazioni che riguardano la selezione dei font) che hanno pregi e difetti; il sottoscritto ha la presunzione di ritenere che l'algoritmo di "long division", come la chiamano gli americani, cioè della divisione che ci hanno insegnato a fare alla scuola elementare, sia l'algoritmo migliore da programmare con TEX.

Nelle pagine 46 e 47 è riportato il listato del pacchetto che contiene tutte le macro che servono per eseguire le tre composizioni e per svolgere la divisione cifra per cifra; le prime poche righe riguardano appunto l'algoritmo della divisione; il comando ad argomenti delimitati `\dividi` (BECCARI, 2008) richiede due lunghezze, una per il dividendo (prima di `\per`) e una per il divisore (dopo `\per` e prima di `\in`) e, infine, il quoziente che viene messo nel terzo argomento (dopo `\in`) e deve essere una sequenza di controllo o un carattere attivo, insomma un solo token che possa ricevere una definizione.

Il lettore non si spaventi se vuole cercare di seguire il processo; in realtà si tratta di assegnare le lunghezze dividendo e divisore a due contatori interi (operazione lecita e in questi contatori va a finire il numero intero di "scaled points" corrispondente a ciascuna lunghezza); dopo una prima

divisione intera fra questi due contatori al fine di determinare la parte intera del quoziente, e dopo avere aggiunto un punto decimale a questa stringa, si entra in una procedura iterativa che moltiplica il resto per 10 ed esegue una divisione aggiungendo il quoziente parziale (di una cifra) alla stringa che contiene il quoziente determinato fino a quel momento; è inutile proseguire oltre la sesta iterazione, perché le cifre eventualmente calcolabili non avrebbero nessun significato quando si consideri il modo di rappresentare internamente le lunghezze adottato da TEX. Se il divisore fosse nullo, al quoziente verrebbe attribuito il valore "infinito", che per TEX, quando si tratta di lunghezze, vale $(2^{30} - 1)/2^{16} \approx \text{\maxdimen}$.

Il resto del pacchetto definisce l'apertura dell'ambiente `\widetable` e della sua chiusura `\endwidetable` con comandi primitivi, perché bisogna leggere e memorizzare in registri token (BECCARI, 2007) sia la larghezza desiderata, sia la descrizione delle colonne, sia il corpo della tabella; questo è definito come tutto quanto sia compreso nel file sorgente fra la fine del gruppo che contiene i descrittori delle colonne e la fine dell'ambiente `\widetable`. Tuttavia non basta fermarsi al comando `\end`, ma bisogna verificare che il suo argomento valga davvero `\widetable`; questo è quello che succede normalmente, ma potrebbe succedere che una cella contenga a sua volta un ambiente non racchiuso fra graffe³, quindi bisogna davvero fare questa verifica e nel caso che l'argomento di `\end` non sia `\widetable` bisogna arrestare la composizione ed emettere un avviso che spieghi il motivo dell'arresto e ponga una tabellina fittizia al posto di quella che la macro non riesce a comporre. Non capita quasi mai una situazione del genere, ma è meglio prevederla.

Merita anche notare che, se la larghezza specificata è di poco inferiore alla larghezza naturale della tabella, $l_0 < l_{\text{spec}} < l_1$, il valore predefinito di `\tabcolsep` ne viene ridotto e la tabella può ancora risultare composta in modo relativamente gradevole; se invece la larghezza specificata è decisamente inferiore alla larghezza naturale, $l_{\text{spec}} < l_0$, `\tabcolsep` viene ad assumere un valore negativo, come si vede chiaramente dall'espressione (1), di modo che le varie colonne risultano parzialmente sovrapposte le une alle altre; certamente non sarebbe difficile inserire un confronto in modo da segnalare l'inconveniente così da evitare questa sgradevole situazione; questo, però, è lasciato per esercizio al lettore. . .

Va anche notato che l'ambiente `\widetable`, così come è stato definito, non consente di specificare il valore facoltativo dell'allineamento verticale della

3. Se il contenuto di una cella di una tabella è racchiuso fra graffe, il suo contenuto viene analizzato solo al momento della composizione, quindi può tranquillamente contenere un altro ambiente, senza che il programma lo isoli come tale.

```

\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{widetable}
\dimendef\wt@Numer=2
\dimendef\wt@Denom=4
\countdef\wt@Num=2
\countdef\wt@Den=4
\countdef\wt@I=6
\def\wt@segno{}

\def\dividi#1\per#2\in#3{%
  \begingroup
  \wt@Numer #1\relax \wt@Denom #2\relax
  \ifdim\wt@Denom<\z@ \wt@Denom -\wt@Denom \wt@Numer -\wt@Numer\fi
  \ifdim\wt@Numer<\z@ \def\wt@segno{-}\wt@Numer -\wt@Numer\fi
  \ifdim\wt@Denom=\z@
    \edef\wt@Q{\ifdim\wt@Numer<\z@-\fi\strip@pt\maxdimen}%
  \else
    \wt@Num=\wt@Numer \wt@Den=\wt@Denom \divide\wt@Num\wt@Den
    \edef\wt@Q{\number\wt@Num.}%
    \advance\wt@Numer -\wt@Q\wt@Denom \wt@I=6
    \@whilenum \wt@I>\z@ \do{\wt@dividiDec\advance\wt@I\m@ne}%
  \fi
% \xdef#3{\wt@segno\wt@Q}\endgroup
\edef\x{\noexpand\endgroup\noexpand\def\noexpand#3{\wt@segno\wt@Q}}
\x
}

\def\wt@dividiDec{%
  \wt@Numer=10\wt@Numer \wt@Num=\wt@Numer \divide\wt@Num\wt@Den
  \edef\wt@Q{\number\wt@Num}\edef\wt@Q{\wt@Q\wt@Q}%
  \advance\wt@Numer -\wt@Q\wt@Denom}

\newdimen\wt@width
\def\wt@starttabular{\expandafter\tabular\expandafter{\wt@preamble}}
\def\widetable#1#2{%
  \def\@tempC{widetable}\setlength{\wt@width}{#1}%
  \def\wt@preamble{#2}\wt@getTable}
\newif\ift@scartare\wt@scartarefalse
\def\endwidetable{%
  \ift@scartare
    \noindent\null
  \else
    \tabcolsep=\z@
    \setbox\z@=\hbox{\wt@starttabular\the\toks@\endtabular}%
    \tabcolsep=1cm\relax
    \setbox\tw@=\hbox{\wt@starttabular\the\toks@\endtabular}%
    \advance\wt@width-\wd\z@
    \@tempdimb=\wd\tw@
    \advance\@tempdimb-\wd\z@
    \dividi\wt@width\per\@tempdimb\in\@tempA
    \tabcolsep=\@tempA\tabcolsep
    \wt@starttabular\the\toks@\endtabular
  \fi
  \ignorespacesafterend
}

\def\wt@finetabella{\end{widetable}}%

```

```

\def\wt@getTable#1\end#2{\def\@tempB{#2}%
  \ifx\@tempB\@tempC
    \toks@={#1}%
    \expandafter\wt@finetabella
  \else
    \PackageWarning{widetable}{%
      La tabella contiene l'ambiente '\@tempB' non chiuso%
    }
    \MessageBreak
    tra graffe. Questo e' espressamente vietato!%
    \MessageBreak
    La tabella non viene composta ma sostituita con%
    un rettangolo}
    \noindent\framebox[\wt@width]{Tabella non composta perch'e
      contiene \texttt{\char'\end} in una o pi'u celle}\par
    \expandafter\wt@finishTable
  \fi
}

\def\wt@finishTable#1\end#2{%
  \def\@tempB{#2}%
  \ifx\@tempB\@tempC
    \wt@scartaretrue\expandafter\wt@finetabella
  \else
    \expandafter\wt@finishTable
  \fi
}
\endinput

```

tabella che ne viene composta; non si è ritenuto indispensabile consentire l'uso di questo parametro facoltativo, benché l'ambiente `tabular*` vi possa ricorrere; non sarebbe complicato eseguire la modifica necessaria per estendere la funzionalità di `widetable`, ma lo si lascia al lettore; lo scopo di questo tutorial, infatti, non è quello di creare un nuovo ambiente, ma di mostrare alcune delle vie che permettono di ovviare agli inconvenienti che \LaTeX presenta quando si compongono tabelle di larghezza specificata.

Questa seconda soluzione potrebbe sembrare ancora più acrobatica di quanto esposto nelle sezioni precedenti; di fatto, copiato il codice in un file dal nome, per esempio, `widetable.sty`, basta invocare questo pacchetto e usare l'ambiente `widetable` al posto di `tabular*` mantenendo esattamente la stessa sintassi di quest'ultimo.

4 Conclusione

Ho voluto qui discutere a livello didattico moderatamente basso i problemi della composizione di tabelle delle quali si vuole specificare la larghezza, e ho indicato alcuni modi per superare gli inconvenienti che si manifestano.

Tutto è migliorabile, ovviamente, comprese le soluzioni che ho indicato; talvolta bisogna ingegnarsi un poco per trovare una soluzione accettabile ai problemi di composizione che \LaTeX non risolve in modo ottimale da solo; questo tutorial vorrebbe sollecitare la fantasia dei lettori a manifestarsi nel-

la ricerca di soluzioni adeguate ai pochi problemi che \LaTeX non riesce a risolvere da solo.

Riferimenti bibliografici

- BECCARI, C. (2007). «I registri *token*: questi sconosciuti». *ArsTeXnica*, (4).
- (2008). «Macroistruzioni con argomenti delimitati». *ArsTeXnica*, (5).
- GUIF (2009). «Introduzione all'arte della composizione tipografica con \LaTeX ». v 0.73.
- KNUTH, D. E. (1996). *The TeXbook*. Addison Wesley, Reading, Mass., 16ª edizione.
- LAMPORT, L. (1994). *A document preparation system — \LaTeX — User's guide and reference manual*. Addison Wesley, Reading, Mass., 2ª edizione.
- PANTIERI, L. (2008). *\LaTeX per l'impaziente — Un'introduzione all'Arte di scrivere con \LaTeX* . Gruppo Utilizzatori Italiani di \TeX e \LaTeX .
- (2009). *L'arte di scrivere con \LaTeX* . URL http://www.lorenzopantieri.net/LaTeX_files/ArteLaTeX.pdf.

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