

I registri *token*: questi sconosciuti

Claudio Beccari

Sommario

In questo tutorial viene mostrato come usare alcuni comandi e oggetti primitivi di $\text{\TeX}$, non accessibili direttamente con $\text{\LaTeX}$, per definire comode macro da usare con (PDF) $\text{\LaTeX}$. Si tratta dei *token* e dei registri *token* che raramente, per non dire mai, vengono trattati nelle guide di $\text{\LaTeX}$.

1 Introduzione

Chi si è letto tutto (e diverse volte) il $\text{\TeXbook}^1$ ha capito che i *token* sono oggetti importanti per l'interprete $\text{\TeX}$, nelle sue varie incarnazioni di “semplice” $\text{\TeX}$, oppure $\text{\PDFTeX}$, $\text{\varepsilon-TeX}$, $\text{\PDF\varepsilon-TeX}$. Siccome questi interpreti servono per tradurre i comandi di alto livello, per esempio le macro di $\text{\LaTeX}$, in comandi primitivi affinché il calcolatore su cui opera svolga il suo compito di comporre un testo nel modo eccellente che tutti conosciamo, è importante sapere che cosa siano i *token* e come gestirli al meglio.

Il $\text{\TeXbook}$ non contiene una definizione precisa introdotta con l'ambiente “definition”, ma espone tante situazioni dalle quali il lettore induce che cosa sia un *token*.

Qui proverò a dare una definizione; per evitare che la definizione si estenda per diversi capoversi, mi limiterò ad una definizione “dogmatica”, sperando che il lettore voglia documentarsi meglio alla fonte, cioè sul $\text{\TeXbook}$.

Un *token* è un oggetto presente nel file sorgente di ingresso che l'interprete acquisisce come se fosse un unico oggetto, anche se è scritto con diverse lettere; i *token* scritti mediante diverse lettere sono costituiti tipicamente dalle sequenze di controllo; a ciascuna di queste è spesso associata una definizione a sua volta costituita da diversi *token*. Queste particolari sequenze di controllo sono *token sviluppabili*; lo sviluppo è costituito dai *token* che formano la definizione. Si chiamano sviluppabili anche i *token* primitivi che eseguono qualche azio-

ne; in questo caso lo sviluppo è il risultato della loro azione.

È chiaro che, se $\text{\TeX}$ legge dei *token* sviluppabili, esso li sviluppa e il loro sviluppo, detto anche *espansione*, sostituisce i *token* originari diventando il nuovo flusso di informazioni che $\text{\TeX}$ deve gestire.

I *token* più frequenti sono le lettere e i segni che il compositore introduce mediante la tastiera nel file sorgente del documento che sta componendo. Ma alcuni *token*, come per esempio la tilde $\sim$, non sono semplici caratteri, ma sono sviluppabili in una sequenza di altri *token*. Tutte le macro composte dal segno di backslash seguito da una sequenza di lettere, oppure da un solo segno che non rappresenti una lettera, sono dei *token* formati da uno o più segni che però l'interprete tratta come un oggetto solo.

Nello stesso tempo non confondiamo i *token* con i gruppi; le cose raggruppate fra parentesi graffe formano un gruppo, possibilmente fatto da diversi *token*; per esempio se si scrive $\text{\tt\{ae\}}$ per ottenere $\text{\ae}$, il file sorgente contiene 5 *token*, precisamente: $\text{\tt\{, a, e, e \}}$; le vocali $\text{\ae}$ sono raggruppate fra parentesi graffe perché esse devono essere elaborate assieme.

La sequenza di controllo $\text{\the}$ è un solo *token*; esso è un comando primitivo e quindi non ha uno sviluppo in termini di altri *token*, ma il risultato della sua esecuzione è costituito dai *token* che rappresentano il contenuto del registro che ne costituisce l'argomento; per esempio, i due *token* $\text{\the\columnwidth}$ restituiscono a $\text{\TeX}$ gli undici *token* che rappresentano la larghezza della colonna corrente: $\text{\tt215.85414pt}$.

Invece la sequenza di controllo $\text{\TeX}$ è inizialmente un solo *token*; siccome esso rappresenta una definizione, cioè è una macro, l'interprete la riconosce come tale e la sostituisce con il suo sviluppo, cioè con la serie di *token* che comparivano nella definizione; in questo caso quella semplice macro viene sostituita con

$\text{\tt T\kern-.1667em\lower.5ex\hbox{E}\kern-.125emX\@}$

Se si conta bene l'unico *token* $\text{\TeX}$ diventa una sequenza di 29 *token*; le sequenze di controllo che compaiono nella definizione di $\text{\TeX}$ sono quasi tutte comandi primitivi, tranne $\text{\@}$, per cui ha luogo una ulteriore sostituzione per sviluppare, appunto, $\text{\@}$.

Se si apre il file $\text{\tt latex.ltx}$ con un editor ASCII, facendo molta attenzione a non apportarvi modifiche (se l'editor lo consente, si apra il file in “read only mode”), e si cerca la stringa $\text{\tt\tokens}$, la si trova

1. Non ritengo necessario inserire nella bibliografia il riferimento al $\text{\TeXbook}$; esso dovrebbe essere la “Bibbia” di qualunque utente del sistema $\text{\TeX}$; parto dal presupposto che ogni lettore ne abbia una copia a portata di mano. Altrettanto realisticamente sono consapevole che questo libro, assolutamente essenziale anche se di difficile lettura, non è fra le letture preferite degli utenti di $\text{\LaTeX}$; questo sia allora un invito a documentarsi meglio facendo riferimento al testo n. 1 da cui discendono tutti gli altri. Non c'è altro modo per capire il comportamento talvolta bizzarro di $\text{\LaTeX}$, e non c'è altro modo di porre rimedio se non con una profonda comprensione del programma che lo fa funzionare.

molto spesso; essa viene usata all'interno del cuore di LATEX per svolgere in modo ottimizzato una quantità di funzioni che sarebbe difficile svolgere diversamente. Più avanti si mostrerà un esempio d'uso e si esamineranno le alternative alla soluzione trovata.

Il problema con i *token* è che la prima operazione che l'interprete esegue leggendo il file sorgente è costituita proprio nel riconoscimento dei *token*; l'operazione è descritta da Knuth come *tokenizzazione*. Se questi *token* devono essere riutilizzati, si rischia di far perdere tempo all'interprete per riconoscere la natura di *token* ad oggetti che ha già riconosciuto e classificato. È per questo che è meglio conservare gli oggetti da riutilizzare (insieme alle loro classificazioni) in opportuni registri, che prendono il nome di *token register*.

Gli utenti LATEX non sono abituati a usare i registri; i pochi che si usano attraverso comandi LATEX sono le scatole (`\newsavebox`), i contatori (`\newcounter`) e le lunghezze elastiche (`\newlength`); altri comandi permettono di eseguire le necessarie operazioni su questi oggetti. Il TEXbook informa il lettore che per ogni categoria di registri se ne possono definire ed usare al massimo 256 (numerati da 0 a 255, senza però che il compositore debba preoccuparsi di gestire questi numeri); l'estensione ϵ -TEX ne può gestire molti di più, ma anche in questo caso il compositore non deve preoccuparsi dei dettagli.

LATEX non contiene nessun dispositivo, nessun comando, per gestire i registri *token*; per usarli bisogna scendere ad un livello più basso di programmazione, ad un livello "primitivo", in quanto occorre servirsi di comandi primitivi.

Tuttavia anche l'utente LATEX un po' avanzato può trarre giovamento dall'uso intelligente dei registri *token*.

2 Un semplice problema: estendere i comandi `\caption` di `babel`

Da quasi 20 anni ho messo a disposizione degli allievi del Politecnico di Torino un pacchetto di macro per comporre le loro tesi; con il variare degli ordinamenti e, ancor di più, con il grande aumento dei nostri allievi che svolgono programmi internazionali di "doppia laurea", è stato necessario integrare quei pacchetti con `babel` estendendone le definizioni alle necessità della composizione della tesi.

Il pacchetto TOPtesi è scaricabile dagli archivi internazionali CTAN, oltre che dal sito ftp del nostro Politecnico; l'ultima versione contiene anche i meccanismi per l'uso di un file di configurazione e per cambiare tutte le *infix string* in modo da poterle rendere in una lingua qualsiasi.

Le *infix string* sono quelle sequenze di caratteri (formanti parole di senso compiuto) inserite direttamente dentro i comandi che eseguono de-

terminate funzioni; per esempio, quando si dà il comando `\caption`, TEX esegue diverse operazioni e a un certo punto scrive la parola "Chapter", oppure "Capitolo", oppure "Chapître", ..., a seconda della lingua in uso. Ebbene le parole "Chapter", "Capitolo", "Chapître", sono sequenze di caratteri contenute, incorporate, dentro la definizione di `\chapter`; in italiano si potrebbero chiamare "stringhe incorporate" ma preferisco usare la locuzione inglese *infix string*.

Mentre il nome "Relatore" deve essere cambiato "a mano" scrivendo una riga opportuna nel file di configurazione, in modo che la stringa diventi "Advisor", "Supervisor", "Superviseur", o quant'altro, altri comandi sono più delicati.

Per esempio esiste un comando `\ringraziamenti` che serve per iniziare una sezione a livello di "capitolo" non numerato, inserita nell'indice generale, con un titolo fisso: "Ringraziamenti" in italiano, "Acknowledgements" in inglese, "Agradecimientos" in spagnolo, eccetera. Il comando non usa una *infix string* per il titolo del capitolo, ma usa una sequenza di controllo che contiene la parola giusta, che cambia automaticamente insieme alla lingua selezionata.

Il comando `\ringraziamenti` ha una definizione che rispecchia molte altre definizioni del genere presenti in moltissime classi di documenti:

```
\newcommand*\ringraziamenti{%
  \iffontmatter\else\frontmattertrue\fi
  \ifopenright\cleardoublepage
    \else\clearpage\fi
  \global\@topnum\z@
  \@afterindentfalse
  \@schapter{\acknowledgename}%
  \addcontentsline{toc}{chapter}%
    {\acknowledgename}%
}
```

Si vede chiaramente che il nome, variabile da lingua a lingua, è contenuto dentro `\acknowledgename`.

Il problema consiste nel "convincere" il comando di `babel` `\selectlanguage` a cambiare anche il nome che rappresenta il titolo di questo capitolo insieme a tutte le altre parole inserite direttamente nei comandi.

Si vorrebbe che ogni utente potesse aggiungere altre definizioni per altre lingue; `babel` consente di comporre in lusazio (!); se mai ci fosse qualche studente che dovesse comporre una tesi in lusazio, dovrebbe essere in grado di definire la parola corretta anche per quella lingua.

3 Una soluzione ottenuta mediante i registri *token*

Il cambiamento delle *infix string* da parte dei comandi di `babel` avviene eseguendo una macro il cui nome si ottiene per agglutinamento della stringa `\captions` con il nome della lingua da usare; ogni

language description file (con estensione `.ldf`), che viene letto in accordo con le varie opzioni di lingua indicate nella chiamata a `babel`, definisce la macro specifica per quella lingua; il file `italian.ldf` contiene la definizione:

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

Analoghe definizioni esistono per le altre lingue².

Noi vorremo costruire un comando L^AT_EX di alto livello (cioè per l'utente finale) che gli consenta di aggiungere la definizione che desidera a qualunque elenco di "captions" in ogni lingua fra quelle che l'utente ha indicato nella lista delle opzioni.

Ecco allora che bisogna scaricare i *token* che formano la definizione di `\captions<lingua>` in un registro *token* e poi usarli per creare un'altra definizione di `\captions<lingua>` con l'aggiunta della nuova definizione. Un comando di assegnazione di una lista di *token* ad un registro *token* numerato *<treg>* si esegue mediante un semplice segno di uguale, ma la lista dei *token* deve essere racchiusa fra parentesi graffe; l'assegnazione pertanto sarà la seguente

```
\toks<treg>=\expandafter{\captions<lingua>}
```

Siccome L^AT_EX non ha un comando per assegnare un nome ad un registro *token*, accontentiamoci di usare il registro 0, sapendo che L^AT_EX è comunque configurato per associare i numeri identificativi dei registri a partire da 10, lasciando cioè i primi 10 (da 0 a 9) per operazioni di tipo scratch. Nello stesso tempo è impossibile far eseguire delle istruzioni dentro ad una lista di *token*; ecco perché compare il comando `\expandafter` prima della graffa aperta; esso dice all'interprete di sostituire il comando successivo alla graffa con la lista dei *token*

2. Il comando `\addto` potrebbe far pensare che basti usare un altro comando simile per aggiungere altre definizioni; la cosa è effettivamente possibile, ma si tratta di un comando "interno" di `babel` ed è meglio lasciarlo stare.

che formano la sua definizione, senza svilupparne nessuno, ma di rimettere la graffa prima della lista in modo che la lista di *token* sia racchiusa fra graffe correttamente appaiate.

Stando così le cose o si conosce il significato di `\captions<lingua>`, oppure l'operazione non può essere fatta; non avremmo nessun problema a scrivere

```
\toks0=\expandafter{\captionsitalian}
```

ma dovremmo mettere questa operazione all'interno della definizione di un comando che si riferisce solo all'italiano; se vogliamo rendere il nostro comando parametrico, dobbiamo prima definire qualcosa di equivalente a `\captionsitalian`, o a `\captionsenglish`, o a `\captionsfrench`, o a...

Ecco allora che torna comodo il comando di equivalenza attraverso uno dei vari giochetti che si possono fare con i comandi primitivi: sia `#1` il parametro che rappresenta la lingua per la quale vogliamo fare l'estensione; dobbiamo creare un unico *token* formato agglutinando `\captions` con il parametro in questione. La soluzione è la seguente

```
\expandafter\let\expandafter\@tempA
\csname_\captions#1\endcsname
```

La spiegazione di questa costruzione è abbastanza semplice: `\let` è un comando che rende due *token* equivalenti; nel nostro caso rende il *token* risultato dell'operazione eseguita mediante la coppia `\csname` e `\endcsname` equivalente al *token* `\@tempA`. Invece la coppia `\csname` e `\endcsname` prende la sequenza di *token* che essa racchiude, eseguendo eventuali sostituzioni se qualche *token* è sviluppabile, e trasforma l'intera sequenza di *token* ottenuta in un'unica sequenza di controllo, quindi in un unico *token*; i due comandi `\expandafter` servono per eseguire gli altri comandi nell'ordine giusto; il primo rimanda al secondo, che a sua volta rimanda a `\csname` che esegue insieme al suo corrispondente `\endcsname` la creazione di un *token* il cui nome è ottenuto agglutinando la parola `captions` con l'argomento che sostituisce il parametro `#1`; eseguita per prima questa operazione l'interprete torna indietro a considerare i due *token* saltati, specificatamente `\let` e `\@tempA`; se `#1` vale `spanish` il gioco congiunto dei comandi suddetti equivale a

```
\let\@tempA\captionsspanish
```

per cui `\@tempA` diventa del tutto equivalente al comando interno di `babel` `\captionsspanish`.

Infine dobbiamo creare la nuova definizione aggiornata del comando `\captions<lingua>`; dobbiamo cioè usare i *token* della vecchia definizione che abbiamo messo nel registro *token* 0 e gli ulteriori comandi che ci servono nel testo della (ri)definizione del comando che ci interessa. Per fare questo abbiamo bisogno di svuotare il registro *token* che

abbiamo appena caricato e lo dobbiamo fare all'interno della nuova definizione; in definitiva, aggiungendo qualche warning adeguato, il nostro nuovo comando `\LTEX` è il seguente:

```
\newcommand*\ExtendCaptions[3]{%
\@ifundefined{captions#1}{%
\PackageWarning{tptesi}%
{Language option #1 not specified
\MessageBreak
Skipping any redefinition\MessageBreak}%
}%
\expandafter\let\expandafter\@tempA
\csname captions#1\endcsname
\toks0=\expandafter{\@tempA%
\def\summaryname{#2}%
\def\acknowledgementname{#3}}%
\expandafter\xdef
\csname captions#1\endcsname{\the\toks0}%
}}%
```

Si noti che il comando `\the` estrae da qualunque registro il suo contenuto; per i contatori `\LTEX` esso non può essere usato, ma c'è il comando alternativo `\value`; per le scatole c'è il comando `\usebox`; per le lunghezze elastiche, invece, e per gli altri registri, `\the` funziona come per `\TeX`. Per i registri *token* `\the` estrae i *token* che esso contiene.

Ora la lettura diventa abbastanza semplice; il nuovo comando `\ExtendCaptions` accetta tre argomenti; il primo è il nome della lingua per la quale si vogliono eseguire i comandi che esso contiene; il secondo e il terzo sono le stringhe che sostituiscono nella lingua specificata le parole italiane “Sommario” e “Ringraziamenti”.

Se si dà il comando

```
\ExtendCaptions{spanish}{Resumen}%
{Agradecimientos}
```

viene esteso il significato di `\captionsspanish` in modo che quando viene eseguito il comando `\selectlanguage{spanish}` anche il titolo del capitolo del sommario e quello dei ringraziamenti vengono correttamente e automaticamente composti in spagnolo.

Il lettore attento potrebbe ora per esercizio ridefinire la macro `\ExtendCaptions` che non si limiti a definire nella lingua giusta le *infix string* solo per il sommario e per i ringraziamenti; potrebbe creare una macro che definisce parametricamente anche la sequenza di controllo che contiene la *infix string* desiderata. Con questa nuova macro si potrebbe aggiungere alle altre definizioni quella di `\definitionname`, per esempio, affinché contenga “Definizione” in italiano, “Definition” in inglese o in francese, “Definición”³ in spagnolo, eccetera. La macro desiderata dovrebbe quindi richiedere tre argomenti: (a) il nome della lingua, (b) la parte della

sequenza di controllo senza **name** che si intende definire, (c) la *infix string* da usare nella definizione. Si veda più avanti un esempio di applicazione di questa macro...

Perché fare tutto ciò quando sarebbe bastato correggere le definizioni all'interno di `spanish.ldf`? Ma perché nessun utente serio e corretto del sistema `\TeX` si azzarderebbe mai a modificare i file di sistema; ogni utente serio e corretto sa infatti che se vuole fare qualcosa di personale, lo può fare tranquillamente copiando e modificando quanto già si trova nei file di sistema, ma lo fa all'interno di propri file contenenti macro personali; al massimo se deve spedire un proprio documento sotto forma di file sorgente, avrà cura di allegare anche il proprio file di macro personali, o almeno una selezione di queste macro, quelle veramente necessarie.

Si noti che nelle definizioni dell'esempio si è fatto uso del carattere `@`; questo implica che le definizioni così come sono non possono essere collocate nel preambolo di nessun documento, a meno di non farle precedere dal fatidico e pericoloso `\makeatletter`; fatidico, perché la maggior parte degli utenti di `\LTEX` se ne dimentica assai spesso; pericoloso, perché togliendo la protezione fornita dal carattere “non lettera”, ma reso temporaneamente “lettera”, è possibile ridefinire per sbaglio qualche comando interno di sistema, col risultato di produrre sfilze di errori incomprensibili e difficili da correggere. Nel mio esempio tutte le sequenze sono controllate; ma se anche non lo fossero, tutto il lavoro eseguito da `\ExtendCaptions` viene eseguito all'interno di un gruppo formato dalle due parentesi graffe che il lettore attento avrà notato, una all'apertura della definizione e l'altra alla chiusura.

Forse si è domandato a cosa possano servire; servono appunto a rendere locali al gruppo le definizioni di tutti i comandi intermedi; l'unico che trascende il gruppo è quello definito mediante `\xdef`. Esso esegue una definizione mediante la sostituzione eventuale di tutti i *token* sviluppabili che compaiono nella definizione, ma rende anche la definizione globale. `\LTEX` manca di comandi equivalenti a `\edef` (definisce espandendo i *token* della definizione) e a `\xdef` (definisce globalmente espandendo i *token* della definizione); per fortuna `\LTEX` manca di questi comandi, perché nelle mani di persone inesperte potrebbero causare danni “irreparabili”.

`\LTEX` è fatto, come tutti ben sappiamo, per comporre documenti composti bene; chi scrive comandi personali, o file di classe, o di estensione, non compone documenti, ma scrive programmi per comporre documenti; solo in questo caso è lecito servirsi di tutti i possibili comandi di basso livello, avendo cura di controllare che tutto si svolga come previsto e che non si dissemini il terreno utilizzato con mine vaganti che regolarmente esplodono nelle mani altrui, oppure anche nelle proprie, ma dopo

3. Si noti che per generalità è necessario scrivere gli accenti presenti nelle *infix string* con la loro corretta sequenza `\TeX`; nel caso di “Definición” bisognerebbe scrivere quindi `Definici\'on`.

diverso tempo, quando non ci si ricorda più dei dettagli del proprio programma.

4 Alternative alla soluzione trovata

È interessante esaminare le alternative, se esistono, alla soluzione trovata.

In versioni precedenti alla 4.00.00 di TOPtesi avevo cercato di risolvere il problema senza ricorrere ai *token*; la soluzione trovata funzionava bene solo con le due lingue di default, l'italiano e l'inglese. Avevo definito due comandi `\italiano` e `\english` che, oltre ad invocare il cambio di lingua fornito dal pacchetto `babel`, (ri)definivano il contenuto delle due stringhe `\summaryname` e `\acknowledgename` in accordo con la lingua scelta. La soluzione era semplice ma non estensibile ad altre lingue arbitrariamente scelte dal laureando; essa non consentiva nemmeno di cambiare le parole che quei comandi inserivano in quelle due *infix string*.

Un'altra alternativa avrebbe potuto essere quella usata dal pacchetto `layout.sty`; questo pacchetto definisce un comando che permette di ottenere il disegno del layout della pagina; esso è un disegno nel quale compaiono il rettangolo del testo, delle testatine e dei piedini, ma vi compaiono anche le varie misure e le varie distanze fra i vari oggetti che compongono quel layout; il pacchetto è molto utile quando si cerca di ridisegnare il layout delle pagine per ottenere non solo una hardcopy di quanto fatto, ma anche per meditarci sopra ed eventualmente apportarvi le necessarie correzioni. Tutte le informazioni verbali riportate sul disegno sono nella lingua prescelta; nella versione inglese comparirà quindi "Header", mentre in quella italiana comparirà "Testatina", eccetera.

Bene, tutte queste serie di *infix string* inserite nei comandi interni che eseguono il disegno sono contenute in sequenze di controllo che vengono attivate con l'opzione scelta nell'invocare il pacchetto; per esempio, invocando il pacchetto con

```
\usepackage[italian]{layout}
```

tutte le informazioni verbali verranno scritte in italiano. Non è però possibile cambiare lingua; il massimo che si può fare è di indicare la lingua scelta fra le opzioni globali della classe del documento e la lingua di default sarà quella che verrà usata nei disegni. Cambiando lingua nel documento, essa cambia ovunque tranne che nel disegno prodotto da `layout.sty`; inoltre non è possibile cambiare le *infix string* di default.

Ora per il pacchetto `layout.sty` non si può dire che la soluzione trovata non vada bene; in fondo si tratta di un pacchetto per generare dell'informazione grafica che serve al compositore, ma che verosimilmente non verrà inserita in nessun documento da rendere di pubblico dominio.

Il pacchetto `varioref.sty`⁴ si trova in una posizione simile; ma la soluzione trovata ricorre ad una particolarità di `babel`; questo pacchetto accetta una serie di *infix string* inserite nella definizione di una macro ottenuta agglutinando la sequenza `\extras` con il nome della lingua, così come lo riceve `babel` nella lista delle sue opzioni; esiste quindi una sequenza `\extrasitalian` che contiene le parole in italiano, una sequenza `\extrasenglish` che contiene quelle in inglese, eccetera. Esistono le traduzioni in molte delle oltre 30 lingue gestibili con `babel`, ma per alcune nessun volontario si è fatto avanti per fornire le traduzioni, per cui le sequenze relative a quelle lingue continuano ad usare le *infix string* inglesi, ma al momento dell'uso viene emesso un messaggio di avvertimento che presenta le "scuse" del pacchetto per non essere in grado di fornire traduzioni adeguate.

Il comando `\selectlanguage` oltre a cambiare i `\captions` delle diverse lingue cambia anche gli `\extras` di quelle lingue, quindi la semplice selezione della lingua da usare attiva anche le *infix string* specifiche del pacchetto `varioref.sty`. Questa soluzione è corretta ed elegante, ma presume che le sequenze `\extras` siano definite per *tutte* le lingue gestibili da `babel`; in ogni caso non consente di ridefinire le stringhe con parole diverse.

Concludendo: la soluzione ottenuta con l'uso dei *token* e di un comando accessibile all'utente quale è `\ExtendCaptions` permette di limitarne l'uso alle sole lingue che veramente interessano nel documento che il compositore sta scrivendo, non richiede "scuse" per le lingue mancanti (perché il compositore difficilmente scriverà in bahasa o in lusazio se non conosce quelle lingue; e se le conosce, sa anche come usare `\ExtendCaptions`) e in più permette di cambiare le stringhe anche per le parole che già vi compaiono di default. Se in italiano il compositore volesse, per esempio, scrivere "Indice generale" invece di "Indice", basterebbe che usasse la forma di `\ExtendCaptions` suggerita per esercizio al lettore semplicemente scrivendo

```
\ExtendCaptions{italian}{contents}%
{Indice Generale}
```

In questo modo verrebbe aggiunta in coda alla lista di definizioni contenute all'interno di `\captionsitalian` la riga

```
\def\contentsname{Indice Generale}
```

e questa seconda definizione di fatto sostituisce la prima, visto che nello scegliere la lingua italiana si chiama implicitamente `\captionsitalian` e quindi vengono eseguite le varie (ri)definizioni in essa contenute nell'ordine in cui sono scritte; se

4. Sia il file `layout.sty` sia il pacchetto `varioref.sty` si trovano nella cartella `/texmf/tex/latex/tools` di qualunque distribuzione del sistema T_EX; i segni di barra diventano di barra rovesciata per i sistemi Windows.

due definizioni si riferiscono alla stessa sequenza di controllo, l'ultima eseguita è quella che vale.

In sostanza la soluzione ottimale dipende dall'uso che si deve fare con le *infix string*; però a me pare che la soluzione ottenuta con l'uso dei *token* e dei registri *token* sia la più versatile.

5 Conclusione

Per usare i registri *token* bisogna servirsi di comandi di basso livello o di comandi primitivi del sistema T_EX. Usando i registri *token* si possono fare cose piuttosto utili, ma è anche possibile addentrarsi in situazioni paludose, come succede sempre quando si usano linguaggi di programmazione troppo “potenti”; sebbene sembri rozzo, il linguaggio

primitivo del sistema T_EX è potentissimo. Quindi: attenzione!

Nonostante questo avvertimento, il lettore non si scoraggi; programmare a basso livello consente di fare cose che i comandi di alto livello generalmente non possono fare; bisogna solo procedere con metodo e controllare sistematicamente che i propri comandi non siano incompatibili con quelli di altri pacchetti, ma, specialmente, che non interferiscano con le macro di sistema.

▷ Claudio Beccari
Dipartimento di Elettronica
Politecnico di Torino
`claudio.beccari@polito.it`