Numero 1
Aprile 2006

ArsTEXnica

Rivista italiana di TEX e LATEX

GUIT

http://www.guit.sssup.it/arstexnica

TEXnica Ars

GUIT – Gruppo Utilizzatori Italiani di TEX

ArsTEXnica è la pubblicazione ufficiale del GUIT

ArsTEXnica è la prima rivista italiana dedicata a TEX, a LATEX ed alla tipografia digitale. Lo scopo che la rivista si prefigge è quello di diventare uno dei principali canali italiani di diffusione di informazioni e conoscenze sul programma ideato quasi trent'anni fa da Donald Knuth.

Le uscite avranno, almeno inizialmente, cadenza semestrale e verranno pubblicate nei mesi di Aprile e Ottobre. In particolare, la seconda uscita dell'anno conterrà gli Atti del Convegno Annuale del GUIT, che si tiene in quel periodo.

La rivista è aperta al contributo di tutti coloro che vogliano partecipare con un proprio articolo. Questo dovrà essere inviato alla redazione di ArsTEXnica, per essere sottoposto alla valutazione di recensori. È necessario che gli autori utilizzino la classe di documento ufficiale della rivista; l'autore troverà raccomandazioni e istruzioni più dettagliate all'interno del file di esempio (`.tex`). Tutto il materiale è reperibile all'indirizzo web della rivista.

Gli articoli potranno trattare di qualsiasi argomento inerente al mondo di TEX e LATEX e non dovranno necessariamente essere indirizzati ad un pubblico esperto. In particolare tutorials, rassegne e analisi comparate di pacchetti di uso comune, studi di applicazioni reali, saranno bene accetti, così come articoli riguardanti l'interazione con altre tecnologie correlate.

Di volta in volta verrà fissato, e reso pubblico sulla pagina web, un termine di scadenza per la presentazione degli articoli da pubblicare nel numero in preparazione della rivista. Tuttavia gli articoli potranno essere inviati in qualsiasi momento e troveranno collocazione, eventualmente, nei numeri seguenti.

Chiunque, poi, volesse collaborare con la rivista a qualsiasi titolo (recensore, revisore di bozze, grafico, etc.) può contattare la redazione all'indirizzo:

`arstexnica@sssup.it`.

Associarsi a GUIT

Fornire il tuo contributo a quest'iniziativa come membro, e non solo come semplice utente, è un presupposto fondamentale per aiutare la diffusione di TEX e LATEX anche nel nostro paese. L'adesione al Gruppo prevede un quota di iscrizione annuale di 25,00 €.

Indirizzi

Gruppo Utilizzatori Italiani di TEX:
c/o Ufficio Statistica
Scuola Superiore Sant'Anna
Piazza Martiri della Libertà, 33
56127 Pisa, Italia.
`http://www.guit.sssup.it`
`guit@sssup.it`

Redazione ArsTEXnica:
`http://www.guit.sssup.it/arstexnica/`
`arstexnica@sssup.it`

Codice ISSN 1828-2369

Stampata in Italia
Pisa: 15 Aprile 2006

ArsTEXnica

Rivista italiana di TEX e LATEX

Editoriale
Massimiliano Dominici 3
Intervista a Donald Knuth
Gianluca Pignalberi 5
I registri *token*: questi sconosciuti
Claudio Beccari 8
Ridefinire i comandi primitivi di TEX e applicazioni a LATEX
Enrico Gregorio 14
L'ambiente picture
Massimo Caschili 20
Norme tipografiche
G. Cevolani 29

Numero 1 Aprile 2006

Stampato a Pisa, Italia, GUIT Gruppo Utilizzatori Italiani di TEX

Editoriale

Massimiliano Dominici

La realizzazione di ArsTEXnica, la prima rivista italiana di TEX, LATEX e tipografia digitale è l'ultima delle iniziative, in ordine di tempo, che la comunità italiana degli utenti di TEX ha messo in campo per diffondere, anche nel nostro paese, la conoscenza e l'uso del programma ideato quasi trent'anni fa da Donald E. Knuth. Un programma che è diventato ormai, a livello internazionale, lo standard *de facto* nel mondo accademico per le pubblicazioni scientifiche (grazie anche al supporto dell'*American Mathematical Society*), ma che è anche anche riuscito ad attrarre, in generale, tutti coloro che sono interessati a produrre documenti di eccellente resa tipografica.

In Italia, bisogna aspettare l'inizio di questo decennio, perché si abbia la presenza di una comunità organizzata di utenti, con un netto ritardo rispetto ad altri paesi. Il TUG (*TEX Users Group*) italiano, infatti, nasce in modo del tutto casuale, ad opera di un gruppo di dottorandi costretti dal loro coordinatore a presentare documenti redatti unicamente in LATEX. Come sempre avviene in questi casi, superata la fase iniziale, rimasti conquistati dalla resa visiva e dalla semplicità del programma, i dottorandi si posero il problema di diffondere, attraverso un sito web corredato da un forum di discussione, informazioni e suggerimenti per coloro che fossero interessati a entrare nel mondo LATEX.

La vera svolta nella storia di questa iniziativa si ebbe nei primi mesi del 2003, quando, grazie al supporto determinante della Scuola Superiore Sant'Anna di Pisa, fu possibile offrire un servizio ancora più professionale ed organizzato. Fu deciso allora di presentare al pubblico, sotto il nome collettivo di GUIT — Gruppo Utilizzatori Italiani di TEX — questa nuova comunità virtuale.

Il supporto e l'impegno profuso ha permesso al GUIT di essere riconosciuto dalla comunità internazionale come TEX Users Group italiano. Ciò ha reso necessario ampliare e riorganizzare, nei contenuti e nella forma, l'intera struttura del Gruppo, passando da un modello di comunità puramente "virtuale" ad una struttura in grado di organizzare eventi di portata nazionale ed internazionale. Questa trasformazione ha permesso quindi il lancio di una serie di iniziative, la più importante delle quali è l'organizzazione dei convegni annuali su TEX, LATEX e tipografia digitale, a livello nazionale.

Tutte queste iniziative, (forum, corsi, convegni), e il loro successo, stanno a testimoniare una crescente vitalità della comunità italiana degli utenti di TEX. Ed è appunto di tale vitalità che la rivista vuole farsi interprete, ponendosi come obbiettivo principale quello di essere il canale privilegiato di comunicazione tra i membri di questa comunità, per tutto ciò che riguarda la diffusione delle conoscenze e la produzione di materiale innovativo.

ArsTEXnica svolgerà questo compito ospitando nelle sue pagine i contributi più interessanti, provenienti non solo dalla comunità italiana, ma anche da autori di lingua diversa. Rivolgendosi ad un pubblico ampio, in cui convivono i principianti assieme ai programmatori più esperti, presenterà materiale adatto alle diverse fasce di interesse: *tutorial* di ogni livello, rassegne e analisi comparate di pacchetti, ma anche studi di applicazioni reali e articoli riguardanti l'interazione con altri programmi e tecnologie correlati (programmi di grafica, XML, linguaggi di programmazione, ecc.).

Questo è ciò che la rivista si propone di fare, ed un assaggio è già presente in questo primo numero. Chi volesse accostarsi per la prima volta alla creazione di semplici figure con LATEX può, ad esempio, consultare l'articolo di Massimo Caschili che spiega come utilizzare a questo scopo l'ambiente picture. Si rivolgono invece ad utenti che desiderano addentrarsi nei meandri della programmazione vera e propria, gli articoli di Claudio Beccari ed Enrico Gregorio. L'aspirante programmatore troverà in questi due articoli materiale preziosissimo su come utilizzare i registri *token* e altri trucchi per ridefinire comandi già esistenti o aggiungervi nuove funzionalità. Gustavo Cevolani, infine, dà una panoramica delle norme tipografiche per la lingua italiana. L'articolo, anche se mirato particolarmente all'applicazione ad un ambiente LATEX, può risultare utile a chiunque utilizzi un programma di videoscrittura per produrre i propri documenti. Inutile dire che l'articolo è risultato utilissimo anche per la composizione di questa rivista.

Naturalmente la rivista non poteva che cominciare con un omaggio al "padre" del TEX, Donald E. Knuth. Gianluca Pignalberi ha tradotto per ArsTEXnica la sua intervista a Knuth, già apparsa in inglese su *Free Software Magazine* e riprodotta su *TUGboat*. Knuth, sollecitato dall'intervistatore, ci rivela i suoi progetti, i suoi interessi, la sua posizione sui brevetti software. Se permettete una piccola anticipazione, questa è la prima di una serie di interviste, realizzate da Gianluca Pignalberi, che avranno come protagonisti i personaggi più importanti nella storia del TEX e dei suoi sviluppi. Non mancate, quindi, la prossima intervista a

Frank Mittelbach, uno degli sviluppatori del *kernel* di LaTeX 2ε.

Per concludere, un ringraziamento va a tutti coloro che hanno lavorato (su base del tutto volontaria, e tuttavia profondendovi un impegno davvero ammirevole) affinché questo primo numero vedesse la luce. Ai redattori, ai recensori, ai revisori di bozze, a tutte le persone che con i loro suggerimenti hanno aiutato a superare le difficoltà man mano che si presentavano, a tutti costoro deve andare un grazie di cuore, da parte di scrive, innanzitutto, ma anche da parte dei lettori. Permettetemi, infine, di esprimere un ringraziamento personale a Maurizio Himmelmann, coordinatore del GUIT, e a Michela Natilli, che mi hanno aiutato a scrivere la parte "storica" di questo editoriale.

Buona lettura.

▷ Massimiliano Dominici
mlgdominici@interfree.it

Un numero primo di domande al Professore Emerito dell'Arte della Programmazione: intervista a Donald Knuth

Gianluca Pignalberi

Sommario

Proponiamo qui la traduzione italiana dell'intervista a Donald Ervin Knuth, apparsa per la prima volta su Free Software Magazine n. 7 e ripubblicata su TUGboat Volume 26, Number 6.

La composizione della rivista Free Software Magazine è interamente basata su TEX. Forse qualcuno non sa ancora che il Professor Donald Knuth ha progettato TEX e l'ha fatto circa 30 anni fa. Da allora il progetto TEX ha generato molti strumenti correlati (ad es., LATEX, ConTEXt, Ω e altri).

Nel 2005 ho avuto l'opportunità e l'onore di intervistare il professor Knuth. Sono orgoglioso, come giornalista e TEXnico di FSM, di vedere l'intervista ripubblicata sulla prima rivista italiana di TEX e LATEX.

Donald E. Knuth, Professore Emerito dell'Arte della Programmazione, professore di matematica (concreta), creatore di TEX e METAFONT*, autore di diversi libri fantastici (come* Computers and Typesetting*,* The Art of Computer Programming*,* Matematica discreta[1]*) e articoli, assegnatario del* Turing Award*, del* Kyoto Prize *e altri importanti riconoscimenti; membro della* Royal Society *(e potrei andare avanti). C'è qualche argomento che avrebbe voluto padroneggiare e non l'ha fatto? Se sì, perché?*

Grazie per le sue parole gentili, ma in realtà provo costantemente ad imparare nuove cose nella speranza di poter poi aiutare ad insegnarle ad altri. Vorrei inoltre poter capire lingue diverse dall'inglese senza troppa difficoltà; sono spesso limitato dalla mia incompetenza linguistica, mentre voglio capire la gente di altre culture ed altre ere.

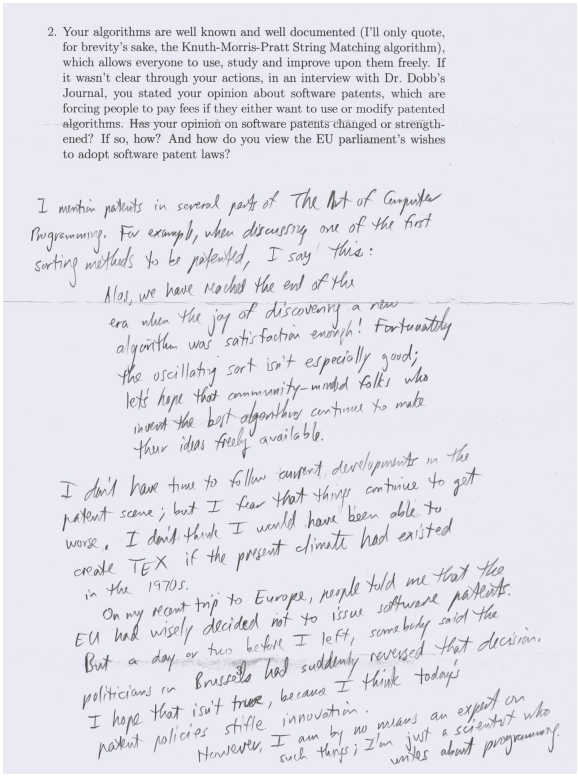
I suoi algoritmi sono ben conosciuti e ben documentati (citerò solamente, per amor di brevità, l'algoritmo di string matching Knuth-Morris-Pratt), cosa che permette a chiunque di usarli, studiarli e migliorarli liberamente. Se non fosse chiaro dalle sue azioni, in un'intervista al Dr. Dobb's Journal ha dichiarato la sua opinione circa i brevetti software, che costringono la gente a pagare delle quote qualora vogliano usare o modificare algoritmi brevettati. La sua opinione sui brevetti software è cambiata o si è rafforzata? In che modo? E come vede i desideri del Parlamento Europeo di adottare leggi sui brevetti software?

Menziono i brevetti in diverse parti di *The Art of Computer Programming*. Per esempio, discutendo uno dei primi metodi di ordinamento ad essere brevettato, dico questo:

> Purtroppo, abbiamo raggiunto la fine dell'era in cui la gioia di scoprire un nuovo algoritmo era una soddisfazione sufficiente! Fortunatamente l'ordinamento oscillante non è particolarmente efficace; speriamo che le persone orientate alla comunità

FIGURA 1: Il prof. Knuth mentre legge una delle riviste composte dal suo programma TEX. Foto di Jill Knuth (laureata al Flora Stone Mather College — un altro FSM)

1. È il poco significativo titolo italiano di *Concrete Mathematics*, dove Concrete è la contrazione di '**con**tinuous' e 'dis**crete**'


2. Your algorithms are well known and well documented (I'll only quote, for brevity's sake, the Knuth-Morris-Pratt String Matching algorithm), which allows everyone to use, study and improve upon them freely. If it wasn't clear through your actions, in an interview with Dr. Dobb's Journal, you stated your opinion about software patents, which are forcing people to pay fees if they either want to use or modify patented algorithms. Has your opinion on software patents changed or strengthened? If so, how? And how do you view the EU parliament's wishes to adopt software patent laws?

I mention patents in several parts of The Art of Computer Programming. For example, when discussing one of the first sorting methods to be patented, I say this:

Alas, we have reached the end of the era when the joy of discovering a new algorithm was satisfaction enough! Fortunately the oscillating sort isn't especially good; let's hope that community-minded folks who invent the best algorithms continue to make their ideas freely available.

I don't have time to follow current developments in the patent scene; but I fear that things continue to get worse. I don't think I would have been able to create TEX if the present climate had existed in the 1970s.

On my recent trip to Europe, people told me that the EU had wisely decided not to issue software patents. But a day or two before I left, somebody said the politicians in Brussels had suddenly reversed that decision. I hope that isn't true, because I think today's patent policies stifle innovation.

However, I am by no means an expert on such things; I'm just a scientist who writes about programming.

FIGURA 2: La porzione di pagina su cui Knuth ha risposto alla seconda domanda di quest'intervista. Prego notate che cita e compone tipograficamente anche quando scrive a mano: la citazione da TAOCP e la parola TEX

> che inventano i migliori algoritmi continuino a rendere le loro idee liberamente disponibili.

Non ho tempo per seguire gli attuali sviluppi nell'ambito dei brevetti; ho paura però che le cose continuino a peggiorare. Non credo che sarei stato in grado di creare TEX se il clima odierno ci fosse stato negli anni 70.

Nel mio recente viaggio in Europa, la gente mi ha raccontato che la UE aveva saggiamente deciso di non imporre i brevetti software. Ma un giorno o due prima che io ripartissi, qualcuno disse che i politici a Bruxelles avevano improvvisamente ribaltato la loro decisione. Spero che non sia vero, perché penso che le odierne politiche dei brevetti soffochino l'innovazione.

Comunque, io non sono affatto un esperto di queste cose; sono solo uno scienziato che scrive di programmazione.

Finora lei ha scritto tre volumi di The Art of Computer Programming*, sta lavorando al quarto, spera di finire il quinto volume entro il 2010, e ancora pianifica di scrivere i volumi 6 e 7. Esclusa la serie* Selected papers*, ci sono altri argomenti di cui lei ritiene di dover scrivere, ma non ne ha avuto il tempo?, Se sì, vuole riassumere di quale argomento parlerebbe?*

Sto facendo progressi lenti ma costanti sui volumi 4 e 5. Ho anche molte annotazioni per i volumi 6 e 7, ma questi libri trattano argomenti meno fondamentali e potrei scoprire che c'è poco bisogno di questi libri quando sarò arrivato a quel punto.

Ho paura dei 20 anni di lavoro necessari prima che io porti TAOCP ad una conclusione di successo. Dato che ho 67 anni ora, spero vivamente di stare in salute ed essere in grado di fare un buon lavoro pur invecchiando ed essendo ancor più decrepito. Fortunatamente, al momento mi sento bene come sempre.

Se avessi tempo per qualcos'altro, mi piacerebbe comporre musica. Naturalmente non so se ciò avrebbe successo; terrei la cosa per me se non fosse buona. Tuttora sento il bisogno di provare ogni tanto e i computer rendono queste cose più semplici.

Circolano voci che lei ha iniziato il progetto TEX perché stanco di vedere i suoi manoscritti maltrattati dall'American Mathematical Society. Allo stesso tempo, afferma di aver scritto TEX dopo aver visto le prove di stampa del libro The Art of Computer Programming*. Per piacere, racconti brevemente ai nostri lettori cosa l'ha fatta decidere ad iniziare il progetto, quali strumenti ha usato, e quante persone facevano parte del nucleo della squadra di TEX.*

No, le società matematiche non potevano essere incolpate per lo stato pietoso della tipografia nel 1975. La cosa è dovuta al fatto che le industrie della stampa erano passate a nuovi metodi e i nuovi metodi erano progettati per essere validi per le riviste, i giornali e i romanzi ma non per la scienza. Gli scienziati non avevano alcun potere economico, così a nessuno importava se i nostri libri e articoli sembravano belli o brutti.

Racconto l'intera storia nel capitolo 1 del mio libro *Digital Typography* che è ovviamente un libro che spero tutti leggano e apprezzino.

Gli strumenti che ho usato sono stati tutti autoprodotti e sono diventati famosi come *Programmazione Colta*[2]. Sono enormemente influenzato dalla programmazione colta che è sicuramente la cosa migliore mai inventata. Io continuo ad usarla per scrivere programmi quasi ogni giorno e mi aiuta ad ottenere codice efficiente, robusto e manutenibile con maggior successo di ogni altro metodo che conosco. Naturalmente, capisco che altre persone potrebbero ritenere altri approcci più di loro gusto; ma, wow, io amo gli strumenti che ho ora. Non avrei potuto affatto scrivere programmi difficili come il metasimulatore MMIX se non avessi avuto la programmazione colta; il compito sarebbe stato troppo arduo.

2. Traduzione letterale di *Literate Programming*, preferita ad altre traduzioni su una mailing list di `linux.it`

FIGURA 3: Ritratto di Donald E. Knuth di Alexandra Drofina. Gli utenti commerciali dovrebbero scrivere a Yura Revich (`revich@computerra.ru`) per il permesso

Nel nucleo della squadra di TEX avevo assistenti che leggevano il codice che scrivevo, che preparavano i driver per le stampanti e le interfacce e li portavano su altri sistemi. Ho avuto due studenti che hanno inventato gli algoritmi per la sillabazione e l'interruzione di riga. E ho avuto molte dozzine di volontari che si riunivano ogni venerdì per diverse ore per aiutarmi a prendere le decisioni. Ma ho scritto da me ogni singola linea di codice.

Il capitolo 10 del mio libro *Literate Programming* spiega perché penso che un progetto di prima generazione come questo sarebbe fallito se avessi provato a delegare il lavoro.

Forse lei ritiene che qualcuna delle tecnologie attuali è ancora insoddisfacente. Se non fosse impegnato a scrivere i suoi capolavori, quale tecnologia proverebbe a rivoluzionare e in che modo?

Be', certamente proverei a lavorare per la pace e la giustizia nel mondo. Tendo a pensare a me stesso come un cittadino del mondo; sono piacevolmente eccitato quando vedo il mondo diventare sempre più piccolo e la gente di diverse culture lavorare insieme rispettando le proprie differenze. Al contrario sono addolorato quando vengo a sapere dell'odio profondo o quando vedo la gente sfruttare gli altri o scacciarli preventivamente.

In che modo può accadere la rivoluzione desiderata? Chissà... ma sospetto che "Ingegneri senza frontiere" siano più vicini di chiunque altro ad una strategia funzionante mediante cui i tecnologi come me possono essere d'aiuto.

Grazie ancora per il suo tempo prezioso.

Grazie per avermi posto domande eccellenti!

▷ Gianluca Pignalberi
Free Software Magazine
`g.pignalberi@`
`freesoftwaremagazine.com`

I registri *token*: questi sconosciuti

Claudio Beccari

Sommario

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

1 Introduzione

Chi si è letto tutto (e diverse volte) il TEXbook[1] ha capito che i *token* sono oggetti importanti per l'interprete TEX, nelle sue varie incarnazioni di "semplice" TEX, oppure PDFTEX, ε-TEX, PDF-ε-TEX. Siccome questi interpreti servono per tradurre i comandi di alto livello, per esempio le macro di 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 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 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 azione; in questo caso lo sviluppo è il risultato della loro azione.

È chiaro che, se 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 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 `~`, 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 `\t{ae}` per ottenere a͡e, il file sorgente contiene 5 *token*, precisamente: `\t`, `{`, `a`, `e`, e `}`; le vocali `ae` sono raggruppate fra parentesi graffe perché esse devono essere elaborate assieme.

La sequenza di controllo `\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* `\the\columnwidth` restituiscono a TEX gli undici *token* che rappresentano la larghezza della colonna corrente: `215.85414pt`.

Invece la sequenza di controllo `\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

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

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

Se si apre il file `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 `\toks`, la si trova

1. Non ritengo necessario inserire nella bibliografia il riferimento al TEXbook; esso dovrebbe essere la "Bibbia" di qualunque utente del sistema 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 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 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 determinate 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{%
 \iffrontmatter\else\frontmattertrue\fi
 \if@openright\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 lingue[2].

Noi vorremo costruire un comando LaTeX 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 LaTeX non ha un comando per assegnare un nome ad un registro *token*, accontentiamoci di usare il registro 0, sapendo che LaTeX 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* 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

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.

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

```
\newcommand*\ExtendCaptions[3]{{%
\@ifundefined{captions#1}{%
\PackageWarning{toptesi}%
{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\acknowledgename{#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 LaTeX 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 LaTeX 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. LaTeX manca di comandi equivalenti a `\edef` (definisci espandendo i *token* della definizione) e a `\xdef` (definisci globalmente espandendo i *token* della definizione); per fortuna LaTeX manca di questi comandi, perché nelle mani di persone inesperte potrebbero causare danni "irreparabili".

LaTeX 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 TEX; 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 TEX. 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 TEX 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

Ridefinire i comandi primitivi di TEX e applicazioni a LATEX

Enrico Gregorio

Sommario

La composizione tipografica eseguita da LaTeX si basa su alcuni comandi primitivi direttamente codificati nel programma TeX. È possibile, sapendo ciò che si fa, *ridefinire* questi comandi. Discuterò alcuni esempi di questo tipo che daranno idee per altre ridefinizioni di comandi.

1 Introduzione

TeX conosce circa trecento comandi *primitivi*, cioè che sono direttamente codificati nel programma. È sulla base di questi che vengono costruite le estensioni a tutti note: prima fra tutte il formato Plain TeX, scritto dallo stesso D. E. Knuth (Knuth, 1984), e il più noto LaTeX, dovuto inizialmente a L. Lamport e poi sviluppato da un gruppo internazionale (Braams *et al.*). In questo articolo assumerò che si lavori in LaTeX.

Il comando primitivo più usato è `\par`. Forse qualcuno dei lettori non ha mai scritto la sequenza di controllo `\par` nei suoi documenti; eppure ha usato questo comando innumerevoli volte: infatti è TeX stesso che si occupa di convertire una riga vuota in un comando `\par`.

Un altro comando primitivo molto usato è `\input`, che ha una sintassi molto particolare: il suo argomento è la più corta stringa di caratteri alfanumerici che lo segue (spazi e caratteri speciali sono esclusi). "Un momento!" dirà più di qualcuno: si deve usare la sintassi

`\input{`⟨*nomefile*⟩`}`

come spiegato in tutti i manuali di LaTeX. Ecco: questo è un esempio di ridefinizione di un comando primitivo.

Il comando `\input` originale ha una sintassi peculiare perché deve adattarsi a vari sistemi operativi. D'altra parte l'autore di LaTeX voleva nascondere all'utente alcune difficoltà e, soprattutto, mantenere il più possibile l'uniformità della sintassi. Un modo per risolvere il problema sarebbe di definire

```
\newcommand{\Input}[1]{\input #1}
```

e quindi obbligare l'utente al comando `\Input`, che avrebbe una forma non usuale (i comandi 'pubblici' di LaTeX sono tutti in minuscolo, tranne alcuni per accenti o simboli).

Si noti l'uso di `\newcommand`, che è specifico di LaTeX; una delle sue funzioni è di controllare che il comando che si vuole introdurre non sia già esistente; se è già definito un comando con quel nome, LaTeX dà un errore. Questo controllo non viene eseguito dal comando primitivo `\def` e dal suo parente `\let`, che vedremo più avanti.

Come fare? La risposta è: ridefinire `\input`. Solo che qualcosa come

```
\renewcommand{\input}[1]{\input #1}
```

è destinato a fallire miseramente. Vedremo fra poco come si risolve il problema.

2 Il carattere `@` e il comando `\let`

In molti comandi 'interni' di LaTeX compare il carattere '`@`'. Questo carattere non può, normalmente, essere usato in questo modo: i nomi dei comandi consistono (1) di una sola "non lettera", (2) di una o più lettere (a–z, A–Z). Tuttavia, quando vengono letti i pacchetti e le classi (`.sty` e `.cls`) LaTeX è in uno stato speciale nel quale il carattere `@` è considerato come una lettera normale; a livello utente si può ottenere lo stesso con il comando `\makeatletter` che va poi annullato con il comando `\makeatother`. I programmatori di pacchetti usano questo trucco per scrivere comandi che non possano essere inavvertitamente modificati. Ogni comando interno ha almeno un carattere `@` nel nome, proprio per questo.

Una delle prime righe di `latex.ltx` (che contiene le definizioni del nucleo di LaTeX) è

```
\let\@@input\input
```

Anche qui vale la convenzione che `@` è una lettera. Che cosa succede? Il comando `\let` seguito da due nomi di comando

```
\let\comandob\comandoa
```

fa in modo che `\comandob` abbia lo stesso significato *attuale* di `\comandoa`; successive ridefinizioni di `\comandoa` non influenzano il significato di `\comandob`. Attenzione: `\let` non esegue controlli come `\newcommand` e quindi è *pericoloso*, se usato da mani non esperte.

Dunque abbiamo a disposizione un comando equivalente a `\input`. Per inciso, questa è una convenzione usata in tutto il nucleo di LaTeX: i comandi che cominciano con due caratteri `@` sono di solito equivalenti a uno dei comandi primitivi. Più in là in `latex.ltx` si trova qualcosa come

```
\renewcommand{\input}[1]{\@@input #1}
```

e questo adesso *funziona*! A dire il vero la definizione è un po' più complicata: non importa, la funzione essenziale che viene eseguita è questa.

Ora un comando come `\input{pippo}` è, agli occhi di TEX, del tutto equivalente a

```
\@@input pippo
```

e, quando raggiunge le profondità del programma dove si eseguono solo comandi primitivi, fa ciò che ci si aspetta: TEX legge il *file* `pippo.tex`.

Un altro comando primitivo ridefinito è `\end`. Il significato di questo comando è 'finisci qui': quando TEX lo incontra, considera chiuso l'ultimo paragrafo ed emette tutte le pagine ancora da comporre (più parecchie altre cose che non sono rilevanti per la discussione).

Lamport decise che la struttura di LATEX fosse ad *ambienti* e che ciascuno di essi fosse delimitato da `\begin{nome}` e da `\end{nome}`. Ovviamente si rese necessario ridefinire il comando primitivo con lo stesso nome. Dunque

```
\let\@@end\end
```

dopo di che siamo liberi di definire `\end` come ci pare, purché ci ricordiamo di usare `\@@end` per dire a TEX di finire il lavoro. E infatti la definizione di `\enddocument` è

```
\def\enddocument{%
...
\@@end}
```

Già che ci sono, spiego a chi non lo sapesse che il comando `\begin{nome}` esegue certe funzioni e alla fine il comando `\nome`, mentre `\end{nome}` esegue altre funzioni di controllo della situazione e il comando `\endnome`. Per questo `\newcommand` emette un messaggio di errore quando si cerca di definire un comando il cui nome cominci con `end`.

Il comando `\def` è quello primitivo di TEX per introdurre nuovi comandi. Il suo uso non è molto difficile, ma va evitato se non si sa quello che si sta facendo: come `\let`, infatti, non esegue alcun controllo sul nome del comando che si sta definendo. Tuttavia è spesso l'unico modo di procedere. Vediamo un esempio, sempre da `latex.ltx`:

```
\def\@setpar#1{\def\par{#1}\def\@par{#1}}
\def\@par{\let\par\@@par\par}
\def\@restorepar{\def\par{\@par}}
```

Che fa il primo comando? Ha un parametro, prima di tutto. Quando viene eseguito, cambia la definizione di `\par` e ridefinisce un comando `\@par` rendendolo equivalente a `\par` (ridefinito).

Il secondo comando definisce `\@par`; quando questa 'incarnazione' di `\@par` viene eseguita, il comando `\par` riprende il suo significato originale (che è ben conservato in `\@@par`) ed eseguito.

Il terzo comando, quando eseguito, fa diventare `\par` un comando che esegue `\@par`.

Confusi? Forse. L'idea è che `\@setpar` venga eseguito dentro un ambiente (nel caso particolare trivlist) e quindi le modifiche che fa a `\par` spariscono alla fine dell'ambiente stesso. Il comando `\@restorepar` viene usato nella definizione di `\vspace` e nelle funzioni di chiusura degli ambienti, per assicurarsi di 'disfare' eventuali modifiche apportate a `\par`.

Usare `\(re)newcommand` in queste situazioni non sarebbe possibile per vari motivi.

L'idea di dare significati particolari a `\par` è stata usata dallo stesso Knuth nelle macro usate per comporre il TEXbook e gli altri volumi della serie *Computers and Typesetting*. Possiamo usare idee simili alle sue per fare una cosa un po' stupida:

```
\makeatletter
\let\ciao@@end\@@end
\def\@@end{\message{^^J^^JCiao, %
  ciao^^J^^J}\ciao@@end}
\makeatother
```

nel preambolo fa in modo che alla fine di una compilazione TEX ci saluti. Lo strano simbolo `^^J` serve a inserire nei messaggi a schermo un fine riga. Provatelo.

Che cosa abbiamo fatto? Semplicemente duplicato la funzione di `\@@end` nel comando `\ciao@@end` (un nome come questo è difficilmente già definito) e poi ridefinito `\@@end` in modo che faccia apparire il messaggio e poi esegua `\ciao@@end` che è del tutto equivalente al comando primitivo `\end` prima di qualsiasi ridefinizione.

3 Il pacchetto everyshi

L'idea di ridefinire un comando primitivo è impiegata nel pacchetto everyshi sul quale sono poi basati prelim2e e eso-pic. Il comando è uno di quelli misteriosissimi, `\shipout`.

Si tratta di un comando che nessuno usa mai esplicitamente: è quello che scarica nel `.dvi` o nel `.pdf` una pagina quando è completa con tanto di materiale annesso (testatina, piè di pagina, note a margine e così via).

Il pacchetto è brevissimo, 31 righe di codice, ma molto efficiente e, soprattutto, indipendente dalle classi che si usano. Infatti la sua azione si svolge dopo che LATEX ha composto una pagina secondo le impostazioni ricevute dalla classe e usa solo comandi di basso livello. Questo permette al pacchetto eso-pic di aggiungere alla pagina, per esempio, le filigrane (in inglese *watermarks*): un bel *Top secret* stampato in grigio sullo sfondo, se si vuole (si veda la documentazione di eso-pic reperibile nella propria distribuzione o su CTAN[1]).

1. CTAN è l'acronimo di 'Comprehensive TEX Archive Network, `www.ctan.org`.

Il pacchetto prelim2e scrive alcune cose nel piè di pagina, utili per identificare le versioni preliminari di un documento.

Un altro uso della ridefinizione di `\shipout` è quello che si impiegava alcuni anni fa, quando le possibilità di agire sui `.dvi` erano limitate e il PDF ancora non esisteva. Un ingegnoso insieme di macro permetteva di stampare due pagine, purché di dimensioni appropriate, sullo stesso foglio e rendeva possibile la produzione diretta di libretti in formato A5. Oppure di risparmiare sui costosi fogli di acetato che servivano a produrre le lastre per la stampa offset[2].

Al giorno d'oggi il problema è molto meno sentito: se si deve stampare un libro si manda al tipografo un documento PDF con le pagine nel formato finale e tutto finisce lì; con il pacchetto geometry è facile definire un formato di 'foglio' arbitrario.

4 Usi veri delle ridefinizioni

Tutto quanto visto prima può sembrare complicato (e lo è); ma ci dà idee per cose che ci possono servire molto più spesso.

Per esempio, vogliamo controllare se nel nostro lungo documento abbiamo abusato del corsivo di enfasi. Come fare perché ci salti agli occhi? Senza però modificare troppo il sorgente e senza dover andare in cerca dei comandi `\emph`.

Ridefiniamo `\emph`, questa è la risposta! Faremo in modo che il nostro corsivo di enfasi appaia in un vistoso rosso che a schermo balzerà evidente:

```
\usepackage{color}
\newcommand{\originalemph}{}
\let\originalemph\emph
\renewcommand{\emph}[1]{%
  \originalemph{\color{red}#1}}
```

Ci sono altri modi per ottenere lo stesso risultato (Beccari, 2006), ma certamente questo è semplice e molto efficace. A che cosa serve la seconda riga? Facile: siccome `\let` non controlla se il comando che segue sia già definito, usiamo `\newcommand` per fare questo lavoro; così, se per caso `\originalemph` fosse già definito, LaTeX reagirebbe con un messaggio di errore.

Supponiamo ora di voler stampare il nostro documento senza che però compaiano le figure inserite con `\includegraphics`; ovviamente vogliamo che lo spazio necessario sia lasciato bianco, quindi l'opzione `draft` alla classe o al pacchetto graphicx non è adatta, perché al posto dell'immagine stampa un riquadro con il nome del *file* grafico.

2. I fogli di acetato sono trasparenti e simili a quelli per le trasparenze; servivano da mastro per l'incisione delle lastre metalliche su cui veniva steso l'inchiostro per la stampa. Fra l'altro, andavano stampati in modo speculare perché il *toner* doveva essere dal lato non a contatto con la lastra, sulla quale ovviamente andava incisa l'immagine speculare della pagina.

Un modo potrebbe essere quello di definirsi un comando `\myig` per fare ciò che si desidera. Ma il documento conterrebbe questo comando e non l'usuale. Si può fare? Certo che si può.

```
\newcommand{\origig}{}
\let\origig\includegraphics
\renewcommand{\includegraphics}[2][]
  {\phantom{\origig[#1]{#2}}}
```

Uso qui la possibilità di definire comandi con un argomento opzionale; nel nostro caso `\includegraphics` viene ridefinito con un argomento opzionale e uno obbligatorio. Ciò che viene sostituito all'argomento opzionale quando non sia specificato è *niente*. Come in precedenza, `\newcommand{\origig}{}` serve a evitare di ridefinire un comando che abbia già un significato.

Esaminiamo ora una chiamata tipica:

```
\includegraphics[scale=.5]{pippo}
```

Questa viene tradotta in

```
\phantom{\origig[scale=.5]{pippo}}
```

Sappiamo che `\phantom` compone ciò che ha come argomento, ma poi lascia solo lo spazio richiesto. Siccome `\origig` ha lo stesso significato originale di `\includegraphics` abbiamo ottenuto quello che desideravamo. Se scriviamo

```
\includegraphics{pippo}
```

questo risulta in

```
\phantom{\origig[]{pippo}}
```

ma a `\includegraphics` originale non dà alcun fastidio avere un argomento opzionale vuoto. *Voilà*. Il documento ha gli stessi comandi che avrebbe normalmente e basterà cancellare le due righe in cui abbiamo definito `\origig` e ridefinito `\includegraphics` per avere un documento che non usa alcun comando strano.

Un problema apparentemente più complicato è quello di colorare tutte le tabelle di verde; precisamente vogliamo che ogni elemento della tabella sia verde, testo e linee. Ovviamente vogliamo mantenere la possibilità di specificare per un ambiente tabular l'argomento opzionale di allineamento e specificare l'ambiente con il suo nome usuale: si tratta di un espediente temporaneo per mettere in evidenza le tabelle.

Qui viene a proposito il modo con cui LaTeX comincia l'ambiente nome, ricordate? Viene eseguita una serie di azioni, l'ultima delle quali è l'esecuzione del comando `\nome`.

```
\newcommand{\origtabular}{}
\let\origtabular\tabular
\renewcommand{\tabular}{%
  \color{green}\origtabular}
```

Che succede quando si chiama tabular dopo aver dato questi comandi? Le solite azioni di inizio ambiente vengono svolte, per finire viene emesso il comando `\tabular` che ora significa

```
\color{green}\origtabular
```

Quindi LaTeX si prepara a scrivere in verde (meglio, istruisce il *driver* a farlo) ed esegue il comando `\origtabular` che ha lo stesso significato dell'originale `\tabular`: questo allora va in cerca dei suoi argomenti al modo solito e quindi la sintassi è a posto.

Si noti che la dichiarazione di 'scrivere in verde' avviene all'interno di un gruppo (che il comando `\begin` ha già aperto); quindi il corrispondente `\end{tabular}` ne annulla l'effetto: fuori da un ambiente tabular il testo sarà nel colore normale.

Funziona tutto sempre? Be', no. Supponiamo di non voler scrivere ogni volta `\hline` dopo ogni riga di una tabella. Sfrutterò ora il fatto che all'interno di un ambiente tabular il comando `\tabularnewline` è equivalente a `\\`. Si potrebbe pensare di scrivere

```
\newcommand{\origtabular}{}
\renewcommand{\tabular}{%
 \renewcommand{\\}{\tabularnewline
   \hline}%
 \origtabular}
```

cioè ridefinire `\\` dopo aver cominciato l'ambiente, per avere ciò che ci aspettiamo. Giusto? No. Provare per credere: nessun messaggio di errore, ma niente linee orizzontali. Perché? Non è così facile: il fatto è che lo stesso comando (originale) `\tabular` chiama un altro comando il quale ridefinisce a sua volta il comando `\\`. E ridefinire questo, che è di quelli che possono avere l'asterisco e un argomento opzionale richiederebbe di ridefinirne un altro e di perdersi nel mare di definizioni: meglio scrivere `\hline`, dopo tutto (o magari evitarlo, in generale).

Ovviamente la strategia di definirsi nuovi ambienti sulla base di ambienti dati rimane utilissima: se per caso abbiamo molte tabelle con un tipo fisso di struttura, è certamente conveniente definire

```
\newenvironment{fixtab}
  {\begin{tabular}{ccc}}
  {\end{tabular}}
```

dove, ovviamente, `ccc` è il tipo di allineamento desiderato. Questo ci permette, per esempio, di modificare la struttura delle colonne di tutte le tabelle di quel tipo agendo in un solo punto del sorgente.

5 Parametri

Molti comandi hanno parametri (e quindi vanno in cerca, quando chiamati, dei loro argomenti). Quando si vogliono ridefinire, questi argomenti possono dare problemi.

Uno dei consigli che vanno maggiormente ascoltati è quello di definirsi *sempre* comandi astratti per le strutture logicamente distinte nel nostro documento. Per esempio, volendo usare notazioni uniformi per gli insiemi numerici (naturali, interi, reali), è bene definirsi una struttura astratta e definire i comandi in termini di quella:

```
\newcommand{\nf}[1]{\mathbf{#1}}
\newcommand{\N}{\nf{N}} % naturali
\newcommand{\Z}{\nf{Z}} % interi
...
```

Ma qui c'è qualcosa di troppo che potremmo eliminare: ci si creda o no, la prima riga può diventare semplicemente

```
\newcommand{\nf}{\mathbf}
```

(o addirittura, se si vuole essere un *macho LaTeX programmer*, `\let\nf\mathbf`). In generale, è meglio evitare di leggere gli argomenti quando non è necessario, useremmo male la memoria di TEX e lo rallenteremmo. Vediamo infatti che cosa succede con la prima definizione quando LaTeX vede `\nf{X}`:

1. per prima cosa si cerca qual è l'argomento; nel caso specifico, `X`;
2. viene sostituito `\nf{X}` con il suo sviluppo, cioè `\mathbf{X}`;
3. ora viene sviluppato `\mathbf` che va in cerca del suo argomento[3].

C'è un passaggio di troppo: irrilevante qui, dove l'argomento è un singolo carattere. Può diventarlo quando si deve valutare un argomento complesso e magari lungo parecchie righe.

Lo sviluppo della seconda definizione di `\nf` evita il problema.

Supponiamo, sempre per esempio, di voler fare in modo che ogni sezione cominci una nuova pagina e, come al solito, di non voler modificare sostanzialmente il sorgente per poter tornare indietro facilmente.

Il comando `\section` è ben diverso da `\includegraphics`: dare un argomento opzionale vuoto non è come non darlo. C'è anche il problema che l'argomento di `\section` è *mobile* e non lo si deve 'agitare troppo' per evitare problemi durante la composizione dell'indice. Per non parlare della possibilità che un asterisco segua il comando.

La soluzione dovrebbe essere ovvia, ormai:

```
\newcommand{\oldsection}{}
\let\oldsection\section
\renewcommand{\section}{%
  \clearpage\oldsection}
```

3. Chi sa di *inner* LaTeX obietterà che `\mathbf` non è veramente un comando con un argomento; di fatto quasi nessuno dei comandi pubblici di LaTeX ha argomenti. Ma, agli scopi pratici, questo è poco importante.

Lasciamo a `\oldsection` il problema di cercare l'asterisco, di vedere se c'è l'argomento opzionale e di valutare l'argomento obbligatorio. Notiamo che questo funziona anche quando si siano caricati pacchetti come titlesec o sectsty che modificano lo stesso comando `\section`, purché la nostra ridefinizione avvenga *dopo* il caricamento del pacchetto e la relativa scelta del formato dei titoli di sezione.

6 Come distinguere le situazioni?

Abbiamo visto due diverse situazioni in cui è necessario adottare strategie diverse per ottenere lo scopo voluto. Se però ci pensiamo, non è così complicato scegliere quella più opportuna.

Nel caso di `\includegraphics` siamo forzati a scegliere di definire un comando con argomenti perché dobbiamo fornire il risultato come argomento di `\phantom`; siamo fortunati perché il comando `\includegraphics` accetta un argomento opzionale vuoto senza alcun problema. Nel caso di `\section`, dobbiamo solo aggiungere qualcosa *prima* di eseguire il comando stesso.

Entriamo ora un po' più a fondo nei meandri del nucleo di LATEX per vedere una diversa applicazione. Come sempre sarà un'applicazione inutile, ma l'idea può forse essere usata per cose più aderenti alla realtà.

Usiamo la classe book; vogliamo che i titoli dei capitoli *non numerati* siano scritti in rosso.

Possiamo ridefinire semplicemente `\chapter`? Forse, ma dovremmo cominciare a vedere se il comando è seguito dall'asterisco oppure no. LATEX permette di definire facilmente comandi con un argomento opzionale, ma non comandi con la forma variata.

Come possiamo agire? Guardiamo `book.cls` e cerchiamo la definizione di `\chapter`. Vedremo che alla fine di quelle righe c'è

```
\secdef\@chapter\@schapter
```

Un po' di intuizione ci dice che `\@chapter` è il comando che viene chiamato quando non c'è l'asterisco. Quello che ci serve allora è `\@schapter`. Un'altra scorsa a `book.cls` rivela che `\@schapter` è un comando con un argomento.

La ricerca è finita: basterà dare

```
\makeatletter
\let\old@schapter\@schapter
\renewcommand{\@schapter}[1]{%
  \old@schapter{\color{red}#1}}
```

Un'applicazione certamente più intelligente è quella di aggiungere automaticamente una riga `\addcontentsline` basata sul titolo del capitolo e la lascio come esercizio al lettore. Una piccola complicazione: vogliamo che l'indice vada nell'indice? Il suggerimento è di rendere attiva una modifica del genere dopo aver composto l'indice e gli elenchi di tabelle e figure.

7 Evitare di leggere un argomento inutilmente

Mi cimento ora con una ridefinizione più complicata di quelle viste prima. Il problema è quello di scrivere in rosso il contenuto di ogni `\mbox`.

Userò il fatto che l'argomento di `\mbox` può essere delimitato anche da `\bgroup` e `\egroup` oltre che da `{` e `}`; inoltre possiamo usare sia un tipo di delimitatore che l'altro. Do subito il codice.

```
\let\redmbox\mbox
\def\mbox{\afterassignment\doredmbox
  \let\next= }
\def\doredmbox{\redmbox\bgroup
  \color{red}}
```

Qui uso `\def` perché sto facendo cose piuttosto complicate e perciò mi servo di comandi a basso livello (come i grandi programmatori che usano l'Assembler). In questo caso `\newcommand` sarebbe andato bene lo stesso.

La procedura è sempre la stessa: si definisce un equivalente del comando da modificare. Il problema qui è che vogliamo entrare nell'argomento di `\mbox`, aggiungere la specifica del colore rosso e poi far eseguire il comando `\mbox` originale sul nuovo argomento.

Entra in gioco di nuovo `\let`. La descrizione della sintassi di `\let` indicata in precedenza non è completa. In realtà

1. `\let` deve essere seguito da un nome di comando;
2. subito dopo può esserci un carattere `=` il quale può essere seguito da uno spazio;
3. infine ci vuole un *token* (vedi Beccari (2006) per sapere che cos'è un *token*).

È perfettamente lecito scrivere `\let\xxx=a`; in tal caso il comando `\xxx` si comporta quasi sempre come il carattere 'a'. Siccome il carattere `=` è opzionale, andrebbe bene anche `\let\xxx a`, ma non sarebbe altrettanto leggibile.

Nel nostro caso il comando `\redmbox`, dopo aver eseguito

```
\afterassignment\doredmbox
```

esegue `\let\next=␣` e quindi fa diventare `\next` equivalente alla parentesi graffa che segue `\mbox`[4].

Che cosa fa `\afterassignment\doredmbox`? Il suo scopo è di eseguire `\doredmbox` *dopo* che è stato eseguito `\let` (che è un comando di assegnazione). Che cosa fa `\doredmbox`? Il suo sviluppo è

4. Qui il carattere `=` è necessario; infatti, nel caso (improbabile) che uno scriva `\mbox=` (che è lecito), LATEX assegnerebbe a `\next` il significato del *token* che segue. È un TEXnicismo, ma è meglio essere precisi; insomma, `\let\next==` assegna a `\next` il significato del carattere `=`, mentre `\let\next=` è ancora *incompleto*.

```
\redmbox\bgroup\color{red}
```

cosicché `\redmbox` 'vede' un argomento che è quello dato ma con aggiunta all'inizio la specifica del colore rosso.

In definitiva, la parentesi graffa aperta che segue `\mbox` viene 'mangiata' con `\let\next={`, quindi TEX vede `\redmbox` che è equivalente a `\mbox` originale, poi la graffa aperta (`\bgroup`) e il comando di scrivere in rosso. È come se scrivessimo ogni volta

```
\mbox{\color{red}...}
```

invece che `\mbox{...}`, che è ciò che volevamo.

Il comando `\next` viene definito, ma non fa niente, se non ingoiare la graffa. È tradizione usarlo proprio per scopi come questo, Knuth lo chiama *scratch macro*, cioè 'macro per scarabocchi', come se fosse un pezzo di carta per prendere appunti.

8 Impacchettare le modifiche

Supponiamo di aver deciso di modificare alcuni comandi a nostro uso durante la composizione (per esempio il rosso per il corsivo). Desideriamo rendere ancora più semplice l'operazione di attivazione o disattivazione delle modifiche.

Il modo più indolore per ottenere lo scopo è di definirsi una propria classe nella quale incorporare queste modifiche. A documento terminato, sarà così necessario solo cambiare *una riga* del sorgente, scegliendo la classe definitiva, diciamo arstexnica.

Il compito sembra imponente, ma vedremo che è semplicissimo. Apriamo un *file* `myarstexnica.cls` nel quale scriveremo

```
\NeedsTeXFormat{LaTeX2e}[1995/12/01]
\ProvidesClass{myarstexnica}
\DeclareOption*{%
 \PassOptionsToClass{\CurrentOption}%
   {arstexnica}}
\ProcessOptions\relax

\LoadClass{arstexnica}
\RequirePackage{color}

\let\myat@emph\emph
\renewcommand{\emph}[1]{%
  \myat@emph{\color{red}#1}}

\let\myat@redmbox\mbox
\def\mbox{\afterassignment\myat@doredmbox
  \let\next= }
\def\myat@doredmbox{\myat@redmbox\bgroup
  \color{red}}
```

A seguire scriveremo il codice delle altre modifiche desiderate e termineremo con `\endinput`. La nostra classe è pronta: non era poi così difficile. Possiamo tenere questo *file* nella stessa cartella dove abbiamo il documento da comporre oppure trasferirlo in una delle aree (locali) dove TEX cerca classi e pacchetti.

Un paio di annotazioni. Le prime due righe sono di identificazione, la terza e la quarta servono a passare le opzioni date alla classe myarstexnica nell'argomento opzionale di `\documentclass` alla classe arstexnica che viene caricata in seguito. La nostra classe privata non definisce alcuna nuova opzione, anche se sarebbe possibile farlo.

La riga successiva è necessaria per elaborare le opzioni. Viene poi caricata la classe sulla quale la nostra si basa e anche il pacchetto color essenziale se vogliamo colorare di rosso qualcosa.

I caratteri '`%`' servono per evitare di lasciare spazi bianchi non voluti nel documento finale; in generale è meglio usarli, quando si scrivono definizioni, nelle righe che non finiscono con un nome di comando. Qui e in altri punti dell'articolo le righe sono spezzate per esigenze tipografiche, nella pratica molte non lo sarebbero.

Ultima nota: quando si scrive una classe o un pacchetto, è conveniente usare come nomi privati di comandi un prefisso seguito dal carattere '`@`'; se il prefisso è scelto bene, sarà improbabile che si ridefiniscano comandi già esistenti. In questo caso ho scelto come prefisso `myat`; a parte la differenza di nomi, la tecnica è quella spiegata in precedenza. Per esempio, `\myat@emph` corrisponde a `\originalemph` della sezione 4.

A questo punto il nostro documento comincerà semplicemente con

```
\documentclass[⟨opzioni⟩]{myarstexnica}
```

e ci basterà cancellare due lettere per ottenere il documento finale.

La classe arstexnica è quella usata per comporre questa stessa rivista, la si può scaricare dal sito di GUIT all'indirizzo `http://www.guit.sssup.it/downloads/arstexnica.zip`

Riferimenti bibliografici

Beccari, C. (2006). «I registri *token*: questi sconosciuti». *ArsTEXnica*.

Braams, J., Carlisle, D., Jeffrey, A., Lamport, L., Mittelbach, F., Rowley, C. e Schöpf, R. «The LATEX 2ε sources». Compreso nella distribuzione di LATEX.

Knuth, D. E. (1984). *The TEXbook*. Addison-Wesley, Reading, MA, USA.

▷ Enrico Gregorio
Dipartimento di Informatica, Università di Verona
Enrico.Gregorio@univr.it

Semplici figure con l'ambiente picture

Massimo Caschili

Sommario

Una breve guida: come usare picture, l'ambiente standard di LaTeX per creare semplici ma efficaci figure.

1 Introduzione

L'ambiente picture delimita uno spazio all'interno del quale è possibile inserire degli oggetti. Gli elementi sono inseriti con precisione usando un sistema di riferimento e le coordinate rispetto a tale sistema. La posizione relativa degli oggetti formerà la figura complessiva. Gli oggetti, inseriti con un comando specifico, possono essere linee, vettori (o frecce), rettangoli, cerchi, ovali, curve o testo semplice. Il disegno, nel suo complesso, è quindi composto assemblando questi oggetti, le cui dimensioni sono espresse in unità. Le modalità d'utilizzo delle unità di misura meriterebbero un articolo dedicato, ma al momento occorre solo sapere che il disegno costruito nell'ambiente picture è proporzionale all'unità di misura indicata dal parametro `\unitlength`. Nel caso non venga specificato, il valore standard adottato è pari a un punto tipografico (1pt, vedi tabella 1) che è possibile (e spesso conveniente) modificare nel seguente modo:

```
\setlength{\unitlength}{1mm}
```

A partire dal punto in cui è stato inserito il comando, l'unità avrà il valore indicato, che, in questo caso, è stata posta pari a un millimetro. Se si decidesse di modificarla, ponendola ad esempio pari a dieci millimetri, tutti gli oggetti all'interno dell'ambiente risulterebbero ingranditi di dieci volte *fuorché il testo*. La modifica del valore di `\unitlength` non ha effetti sul testo.

2 Dichiarazione dell'ambiente

L'ambiente ha la seguente sintassi:

```
\begin{picture}(Δx, Δy)(x_a, y_a)
⋮
Oggetti
⋮
\end{picture}
```

Gli argomenti Δx e Δy indicano rispettivamente la larghezza e l'altezza dell'area che conterrà il disegno, espressi nelle unità correnti, mentre le coordinate x_a e y_a individuano l'angolo inferiore sinistro dell'area delimitata dall'ambiente rispetto all'origine delle coordinate. Se si pone `(0,0)`, l'origine coincide con l'angolo inferiore sinistro dell'area: omettere (x_a, y_a) equivale a porre `(0,0)`.
Esempio:

```
\begin{picture}(50,40)(0,0)
⋮
\end{picture}
```

Questo codice delimita un'area con le caratteristiche evidenziate in figura 1. Le linee tratteggiate[1] delimitano il perimetro dell'area con larghezza 50 unità ed altezza 40 unità. Si noti la posizione dell'origine delle coordinate (0,0): l'angolo superiore destro avrà di conseguenza coordinate (50,40).

in	$1\ in = 2,54\ cm$
	$1\ in = 72,27\ pt$
pt	$1\ pt = 0,3515\ mm$
bp	$1\ in = 72\ bp$
pc	$1\ pc = 12\ pt$
dd	$1157\ dd = 1238\ pt$
	$1\ dd = 0,3759\ mm$
cc	$1\ cc = 12\ dd$
em	Misura relativa pari all'altezza della lettera M del carattere corrente
ex	Misura relativa pari alla larghezza della lettera x del carattere corrente

Tabella 1: Unità di misura

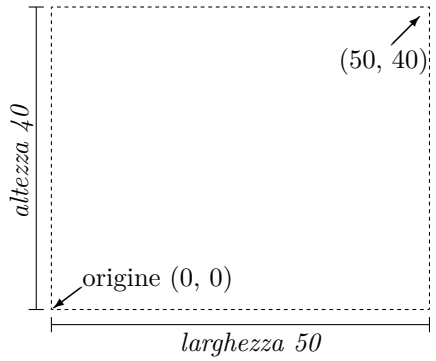


Figura 1: Area delimitata dall'ambiente picture

Più esattamente il codice è:

1. Il tratteggio è usato solo per evidenziare l'area definita dall'ambiente, che in realtà, ha bordi invisibili.

```
\begin{center}
\setlength{\unitlength}{1mm}
\begin{picture}(50,40)(0,0)
   oggetti
\end{picture}\end{center}
```

L'ambiente center centra la figura e, nella seconda riga, si fissa ad un millimetro l'unità di misura per l'ambiente `\picture`.

Proviamo ora a cambiare le coordinate dell'angolo inferiore sinistro, portandole a (5,5) come in figura 2.

L'area delimitata ha le stesse dimensioni di prima, come mostra il codice seguente usato per generarla:

```
\begin{picture}(50,40)(5,5)
⋮
\end{picture}
```

Il punto di coordinate (0,0) risulterà fuori dall'area 'protetta' dall'ambiente. Ciò significa che è possibile posizionare gli oggetti in qualunque punto della pagina rispetto all'area individuata dall'ambiente, come ad esempio il punto di coordinate (-19,0); ovviamente ciò comporterà possibili sovrapposizioni con altri elementi della pagina.

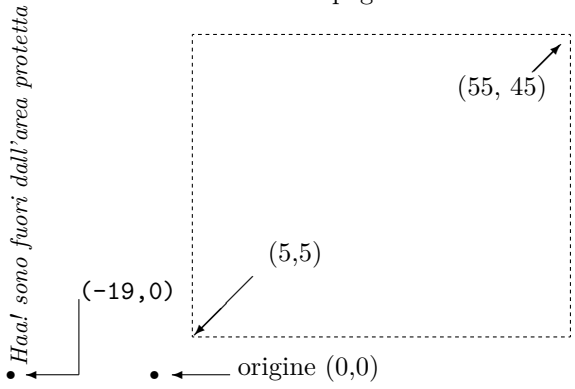


FIGURA 2: Origine diversa da (0,0) e oggetti fuori area

Per inserire gli oggetti nell'ambiente si usa il comando `\put`, la cui sintassi è:

`\put(x,y){`*oggetto*`}`

(x,y) Indicano le coordinate in cui inserire l'oggetto.

oggetto Indica uno qualunque degli oggetti dell'ambiente, l'oggetto può anche essere del semplice testo.

Per inserire più volte il medesimo oggetto:

`\multiput(x,y)(Δx,Δy){`*n*`}{`*oggetto*`}`

(x,y) Indicano le coordinate dalle quali parte l'inserimento multiplo dell'oggetto.

(Δx,Δy) Indicano la distanza (orizzontale Δx, e verticale Δy) che deve separare un oggetto dall'altro.

n Indica quante volte deve essere ripetuto l'inserimento, cioè quanti oggetti inserire.

oggetto Indica uno qualunque degli oggetti dell'ambiente, o un testo.

Inserire un oggetto significa mettere *il punto di riferimento* dell'oggetto in coincidenza delle coordinate indicate. È bene chiarire cosa sia questo *punto di riferimento*: ogni oggetto ha il suo e sarà indicato nella sezione dedicata (es. il punto di riferimento di un rettangolo è l'angolo inferiore sinistro). Per quanto concerne il punto di riferimento dei caratteri è bene far riferimento alla figura 3; si noti la posizione del punto di riferimento per i caratteri `G` e `g`. I punti di riferimento dei caratteri della stessa parola cadono tutti quanti sulla *linea di base*.

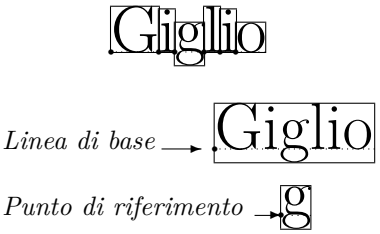


FIGURA 3: Linea di base e punto di riferimento

Semplificando, possiamo dire che TEX e LATEX trattano i caratteri come oggetti inseriti in scatole: per ogni parola sarà costruita un'unica scatola contenente la sequenza delle scatole relative a ciascuna lettera che forma la parola.

`\put(35,45){Testo}`

`(35,45)` → Testo

`\multiput(20,10)(10,10){3}{X}`

X

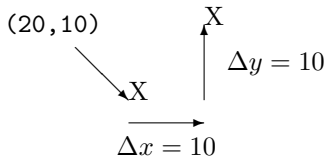


FIGURA 4: Esempio dell'uso di `\put` e `\multiput`

3 Rettangoli

Un rettangolo[2] può essere prodotto con uno dei seguenti comandi:

2. Nei manuali in inglese si parla genericamente di *box*, cioè scatola.

Opzioni	Significato
t	Alto, sul limite superiore del rettangolo, centrato rispetto alla larghezza.
b	Basso, sul limite superiore del rettangolo,centrato rispetto alla larghezza.
l	Sinistra, sul limite sinistro del rettangolo,centrato rispetto all'altezza.
r	Destra, sul limite destro del rettangolo,centrato rispetto all'altezza.
s	Estende il testo orizzontalmente per occupare tutta la lunghezza della scatola, centrato verticalmente.
tl	Alto e a sinistra, inserisce il testo presso l'angolo in alto a sinistra.
tr	Alto e a destra, inserisce il testo presso l'angolo in alto a destra.
bl	Basso e a sinistra, inserisce il testo presso l'angolo in basso a sinistra.
br	Basso e a destra, inserisce il testo presso l'angolo in basso a destra.

TABELLA 2: Le opzioni per i comandi `\makebox`, `\framebox`, `\dashbox` indicano in quale posizione, relativa al rettangolo sarà inserito il testo.

`\makebox(`$\Delta x, \Delta y$`)[opzioni]{ogg.}`
`\framebox(`$\Delta x, \Delta y$`)[opzioni]{ogg.}`
`\dashbox{ltratt}(`$\Delta x, \Delta y$`)[opzioni]{ogg.}`
Le opzioni sono riportate nella tabella 2.

Il comando `\makebox` crea un rettangolo con perimetro invisibile.

Il comando `\framebox` crea un rettangolo con perimetro continuo.

Il comando `\dashbox` crea un rettangolo con perimetro tratteggiato e lunghezza del tratteggio pari a `ltratt`; Δx e Δy definiscono rispettivamente la larghezza e l'altezza del rettangolo.

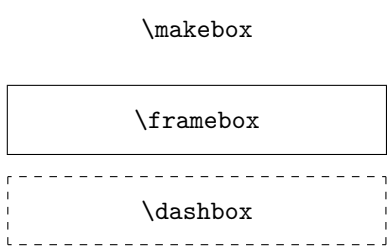


FIGURA 5: Tipi di rettangoli con relativo comando, hanno tutti le stesse dimensioni, il primo ha i bordi invisibili

Il comando `\shortstack` è un po' particolare: crea un rettangolo con bordi invisibili e opera come una tabella a colonna singola. Il testo, accettato come argomento, può essere disposto su più righe separate dal comando `\\`. L'indicazione riguardo

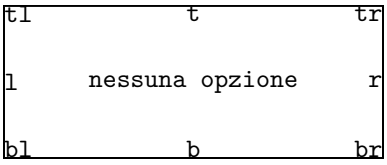


FIGURA 6: La disposizione delle lettere corrisponde alla disposizione data dalle relative opzioni.

l'allineamento del testo è opzionale e lo stile predefinito è quello centrato. Per ottenere un allineamento a destra o a sinistra si usa rispettivamente `r` e `l`. La sintassi completa del comando è:

`\shortstack[op]{ogg \\ ... \\...}`

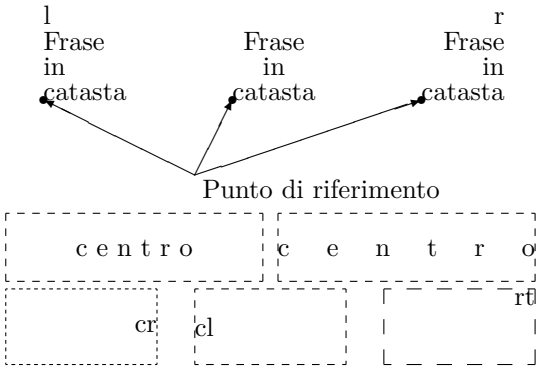


FIGURA 7: In alto esempi per il comando `\shortstack` (si notino i punti di riferimento). In basso risultato del comando `\dashbox`, senza opzione, con opzione `s`, con tratteggio di 0,5 1 e 2 millimetri, il testo indica l'opzione usata per posizionarlo.

4 Linee e frecce

L'oggetto *linea* (che in realtà è un segmento) si ottiene con il comando `\line`, mentre l'oggetto *freccia* (o *vettore*) si ottiene con il comando `\vector`. La sintassi per questi due comandi è:

`\line(`$\Delta x, \Delta y$`){lung}`

`\vector(`$\Delta x, \Delta y$`){lung}`

Gli incrementi Δx e Δy sono rappresentati da valori interi che vanno da -6 a $+6$ per `\line` e da -4 a $+4$ per `\vector`; `lung` è l'incremento della variabile x che individua l'ascissa del secondo estremo del segmento.

Vediamo un esempio in figura 8.

Il segmento AB è stato tracciato con il seguente codice:

`\put(10,10){\line(2,1){35}}`

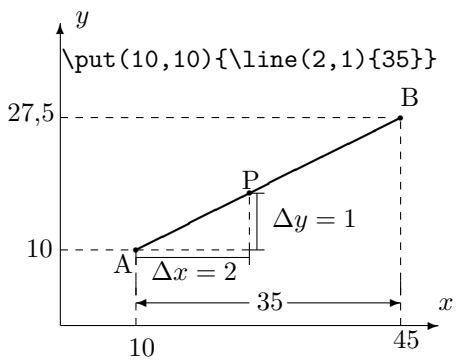


FIGURA 8: Oggetto linea e il codice relativo

Il comando `\put(10,10)` fissa il punto d'inserimento dell'oggetto `\line`, di coordinate `(10,10)`, che indichiamo con A. L'istruzione `{\line(2,1){35}}` traccia il segmento e il suo argomento, `(2,1)`, indica di quanto incrementare le variabili per dare al segmento una specifica inclinazione. In pratica, aumentando di 2 la variabile x (mi sposto verso destra di due unità) e di 1 la variabile y (mi sposto verso l'alto di una unità), si individua il generico punto 'successivo' del segmento. Dato che per due punti passa una e una sola retta, è così determinata l'inclinazione del segmento. Aumentando la variabile x di 35 unità, si determina la lunghezza del segmento e il punto B avrà ascissa $x = 10 + 35$ (l'ordinata risulterà $y = \frac{35}{2} = 17,5$).

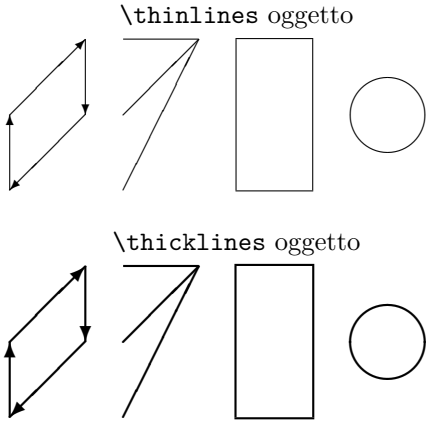


FIGURA 9: Linee e vettori con spessore `\thinlines`, in alto, e `\thicklines`.

Gli elementi `\line`, `\vector`, `\circle`, `\oval` possono essere disegnati anche con linee un più spesse, anteponendo il comando `\thicklines` al comando che produce l'elemento. Infatti, da quel punto in poi fino alla chiusura dell'ambiente, tutti gli oggetti saranno disegnati con una linea più spessa, a meno d'inserire il comando `\thinlines` per riportare lo spessore delle linee allo stato predefinito (più sottile).

Il comando `\linethickness{spess}` stabilisce lo spessore delle linee con `spess` qualunque, che deve essere indicato con l'unità di misura; `\linethickness` ha effetto solo su `\vector`, `\line` (a patto che siano in posizione orizzontale o verticale), sugli oggetti `\framebox`, `\dashbox` e sulle curve ottenute con `\qbezier`. Bisogna fare attenzione all'uso combinato di `\linethickness{spess}` e `\vector`, infatti, all'aumentare di `spess`, s'inspessisce il segmento ma non la punta della freccia (che risulta *inglobata* già con spessori modesti della linea). In tal caso, vista la resa scadente, se ne sconsiglia l'uso. Per la precisione, `\linethickness` non ha effetto su `\circle` e `\oval`.

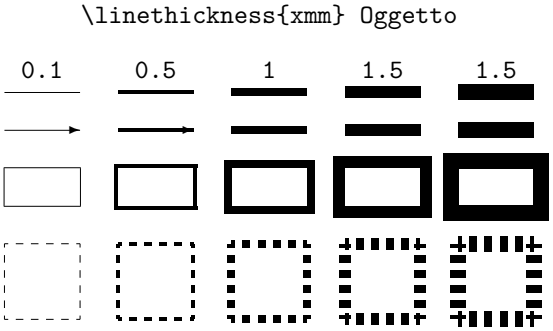


FIGURA 10: Linee e rettangoli con diverso spessore, si noti la pessima resa sul comando `\vector` e, per spessori notevoli, con `\dashbox`.

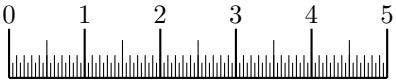


FIGURA 11: Una semplice applicazione del comando `\multiput` e `\line`, vedi il codice nella sezione 9.

5 Circonferenze e cerchi

Gli oggetti circonferenza e cerchio possono essere tracciati, rispettivamente, con i seguenti comandi:

```
\circle{diam}

\circle*{diam}
```

per i quali la misura del diametro è espressa nell'unità corrente. Il diametro massimo consentito per `\circle` è di 40pt, mentre il disco massimo riproducibile con `\circle*` ha diametro pari a 15pt. Se si utilizzano unità diverse dal punto tipografico, è bene operare la conversione nell'unità di misura che si intende usare, vedi tabella 1. Il centro del cerchio rappresenta il punto di riferimento dell'oggetto, si veda la figura 12.

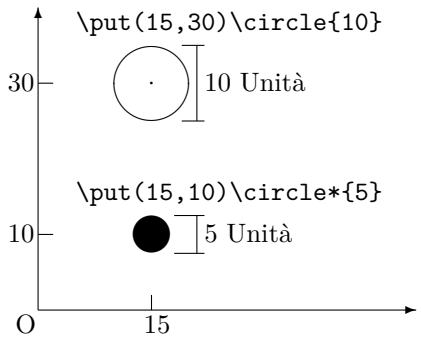


FIGURA 12: Si noti che il punto di riferimento è il centro. Non tutte le misure del diametro sono possibili, il diametro massimo per `\circle` è di 40pt, per `\circle*` il massimo è 15pt

6 Angoli arrotondati

Il comando `\oval`, a dispetto del nome, non costruisce veri e propri ovali o ellissi, ma piuttosto dei rettangoli con angoli smussati. L'effetto è ottenuto usando quarti di cerchio raccordati con segmenti, al limite di lunghezza nulla; la massima curvatura dei quarti di cerchio è pari a `\circle` con diametro massimo. La sintassi del comando è la seguente:

`\oval(`Δx`,`Δy`)[op]`

Le opzioni `op` permettono di disegnare parti dell'oggetto e se tali opzioni si omettono, come mostrato in figura 14, la figura sarà disegnata per intero. Si noti che in figura 13 il punto di riferimento dell'ovale coincide con il *centro* della figura intera.

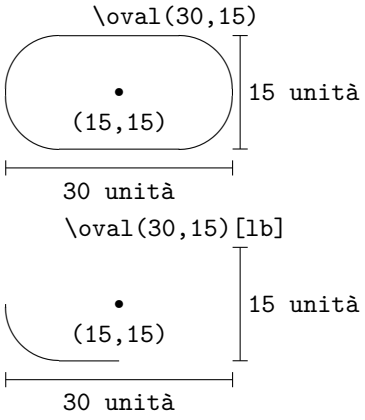


FIGURA 13: In alto, `\oval` è inserito con `\put(15,15)` e manca l'opzione, per cui la figura è intera; sotto, l'opzione è indicata e traccia solo la porzione inferiore sinistra dell'oggetto. Si noti che le dimensioni dei due oggetti, sono le stesse così come è la stessa la posizione relativa del punto di riferimento.

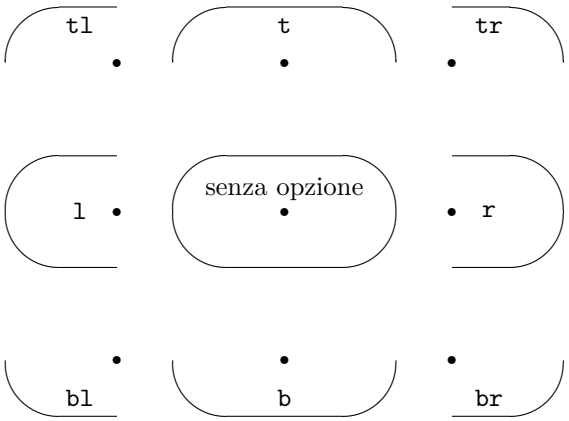


FIGURA 14: Panoramica delle opzioni per `\oval`

7 Curve

Disegnare curve generiche non è cosa da poco. Il comando `\qbezier` consente di disegnare curve sotto forma di archi individuati da tre punti (vedi figura 15), dove i segmenti sono tangenti alla curva. La sintassi del comando è:

`\qbezier(`x_1, y_1`)(`x_2, y_2`)(`x_3, y_3`)`
`\qbezier[n](`x_1, y_1`)(`x_2, y_2`)(`x_3, y_3`)`

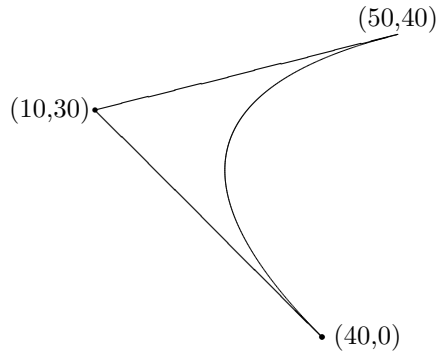


FIGURA 15: Curva `\qbezier` e relative tangenti

Se l'opzione è omessa, il tratto della curva risulterà continuo, altrimenti viene descritta una curva con un numero di punti pari a `n`. Il primo e il terzo punto fissano gli estremi della curva mentre il secondo punto è quello d'intersezione delle tangenti alla curva nei punti uno e tre. Se i tre punti `(`x_1, y_1`)(`x_2, y_2`)(`x_3, y_3`)` sono allineati, il risultato è rappresentato da una retta.

8 Ambienti annidati

Come si è visto, l'ambiente picture genera un'area le cui dimensioni sono specificate al momento della dichiarazione d'apertura dell'ambiente. Quest'area è a tutti gli effetti una *scatola* come le altre trattate da LaTeX, ma occorre fare attenzione ad individuare il suo punto di riferimento che chiamiamo `(`x_a, y_a`)`. Esso, infatti, coincide con l'angolo

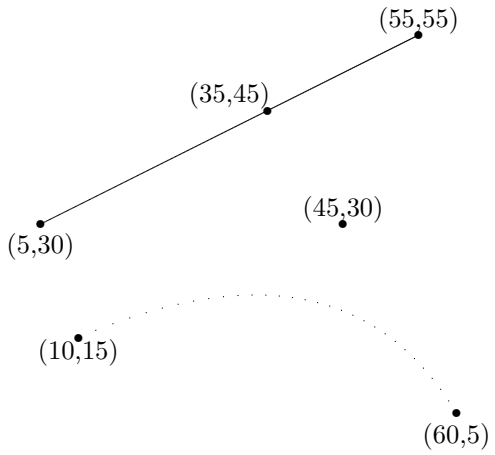


FIGURA 16: Curva `\qbezier` con opzione n = 30, e in alto il risutato con tre punti allineati, in evidenza i punti di riferimento.

inferiore sinistro solo nel caso di `(`x_a, y_a`) = (0,0)`. Detto ciò, è possibile annidare un ambiente picture all'interno di un altro e la posizione indicata da `\put(`x_i, y_i`)` inserirà l'ambiente in modo che `(`x_a, y_a`) = (`x_i, y_i`)`. L'esempio di figura 17 è costruito inserendo due ambienti all'interno di un terzo: in ciascuno degli ambienti è stato inserito un rettangolo di dimensioni uguali all'area dichiarata dall'ambiente stesso.

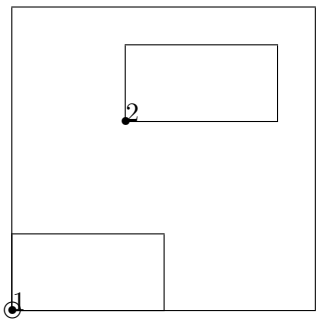


FIGURA 17: Ambienti annidati, tutti gli oggetti sono stati inseriti in `(0,0)`, si veda il codice relativo.

Codice figura 17

```
\begin{center}
\setlength{\unitlength}{1mm}
\begin{picture}(40,40)(0,0)
 \put(0,0){\framebox(40,40){}}
 \put(0,0){\circle{2}}
 \put(0,0){\begin{picture}(20,10)(0,0)
            \put(0,0){\framebox(20,10){}}
            \put(0,0){\circle*{1}}
            \put(0,0){1}
           \end{picture}}
 \put(0,0){\begin{picture}(20,20)(-15,-25)
            \put(0,0){\framebox(20,10){}}
            \put(0,0){\circle*{1}}
            \put(0,0){2}
           \end{picture}}
\end{picture}
```

Come si legge alla riga 6 `(`x_a, y_a`) = (0,0)`, questo ambiente è inserito esattamente all'origine delle coordinate con `\put(0,0)`. Il secondo ambiente (riga 11) risulterà traslato perché `(`x_a, y_a`) = (`$-15, -25$`)`. L'annidamento degli ambienti semplifica la costruzione di disegni complessi, costruendo i singoli pezzi in ambienti distinti per poi assemblarli alla fine. Questa tecnica è stata usata per realizzare la figura 13, come indicato dal codice seguente:

Codice figura 13

```
\begin{picture}(45,55)(0,0)
 \put(22.5,53){\makebox(0,0){
       \texttt{\char`\\oval(30,15)}}}
 \put(0,0){\circle*{1}}
 \put(0,28){
   \begin{picture}(45,25)(0,0)
    \put(15,15){\oval(30,15)}
    \put(15,15){\circle*{1}}
    \put(0,5){\line(1,0){30}}
    \put(0,4){\line(0,1){2}}
    \put(30,4){\line(0,1){2}}
    \put(15,11){\makebox(0,0){%
       \texttt{(15,15)}}}
    \put(15,2){\makebox(0,0){%
       \texttt{30 unità}}}
    \put(32,14){\texttt{15 unità}}
    \put(31,7.5){\line(0,1){15}}
    \put(30,7.5){\line(1,0){2}}
    \put(30,22.5){\line(1,0){2}}
   \end{picture}}
  \put(22.5,25){\makebox(0,0){
        \texttt{\char`\\oval(30,15)[lb]}}}
  \put(0,0){
   \begin{picture}(45,25)(0,0)
    \put(15,15){\oval(30,15)[bl]}
    \put(15,15){\circle*{1}}
    \put(0,5){\line(1,0){30}}
    \put(0,4){\line(0,1){2}}
    \put(30,4){\line(0,1){2}}
    \put(15,11){\makebox(0,0){%
       \texttt{(15,15)}}}
    \put(15,2){\makebox(0,0){%
       \texttt{30 unità}}}
    \put(32,14){\texttt{15 unità}}
    \put(31,7.5){\line(0,1){15}}
    \put(30,7.5){\line(1,0){2}}
    \put(30,22.5){\line(1,0){2}}
   \end{picture}}
\end{picture}
```

9 Esempi e codice

In questa sezione sono presentati diversi esempi, con relativo codice commentato, e il codice di alcune figure dell'articolo.

Leggiamo il codice che riproduce la figura 11; la riga 1 apre l'ambiente center, che centrerà orizzontalmente la figura, e che si chiude con la riga 15. La riga 2 dichiara un contatore di nome num, il cui valore predefinito è zero e sarà usato nella riga 9. Nella riga 3 si fissa la dimensione dell'unità di misura. Con la riga 4, l'ambiente costruirà un rettangolo privo di bordi di 50 per 15 millimetri. L'angolo inferiore sinistro dell'area coincide con l'origine delle coordinate e nel caso in questione l'indicazione facoltativa (0,0) è stata esplicitata. Con le righe 5-6-7 s'inseriscono i segmenti per le tacche di uno, mezzo e cinque millimetri. La riga 8 indica d'inserire sei volte quanto eseguito dal suo argomento, che si chiude nella riga 10; \arabic{num} specifica che sarà mostrato il valore del contatore num in cifre arabe. Tale contatore ha valore iniziale pari a zero e, di volta in volta, viene incrementato di una unità tramite \addcounter{num}{1} (riga 10). Infine \multiput ripete l'inserimento del valore del contatore. Si noti che, per centrare al meglio i numeri, essi vengono racchiusi uno ad uno (riga 9) all'interno di scatole di dimensione zero in altezza e larghezza. Si realizza ciò per far sì che il loro punto di riferimento coincida con il *centro* del numero, il quale sarà posto nel punto indicato. La riga 11 dichiara \thicklines, col risultato d'inspessire la linea di tracciatura degli oggetti che seguono (vedi anche il codice della figura 9).

Codice del righello di figura 11

```
1 \begin{center}
2   \newcounter{num}
3   \setlength{\unitlength}{0.1cm}
4 \begin{picture}(50,15)(0,0)
5     \multiput(5,0)(10,0){5}{\line(0,1){5}}
6     \multiput(1,0)(1,0){49}{\line(0,1){3}}
7     \multiput(0.5,0)(1,0){50}{\line(0,1){2}}
8     \multiput(0,8.5)(10,0){6}{%
9        \makebox(0,0){\arabic{num}}
10       \addtocounter{num}{1}}
11  \thicklines
12    \put(0,0){\line(1,0){50}}
13    \multiput(0,0)(10,0){6}{\line(0,1){6.5}}
14 \end{picture}
15 \end{center}
```

Il codice che riproduce la figura 9 mostra l'uso dei comandi \thinlines (riga 4) e \thicklines (riga 15). Il significato di tali comandi è immediato: il primo, una volta dichiarato, influenza il codice che lo segue fino alla dichiarazione del secondo. I due comandi sono usati all'interno di un ambiente, pertanto essi hanno dominio solo al suo interno.

Codice figura 9

```
1   \begin{picture}(45,58)(0,0)
2     \put(22.5,53){\makebox(0,0){
3       \texttt{\char`\\thinlines} oggetto}}
4    \thinlines
5     \put(0,30){\vector(0,1){10}}
6     \put(0,40){\vector(1,1){10}}
7     \put(10,50){\vector(0,-1){10}}
8     \put(10,40){\vector(-1,-1){10}}
9     \put(15,30){\line(1,2){10}}
10    \put(15,40){\line(1,1){10}}
11    \put(15,50){\line(1,0){10}}
12    \put(30,30){\framebox(10,20){}}
13    \put(22.5,23){\makebox(0,0){
14      \texttt{\char`\\thicklines} oggetto}}
15  \thicklines
16    \put(0,0){\vector(0,1){10}}
17    \put(0,10){\vector(1,1){10}}
18    \put(10,20){\vector(0,-1){10}}
19    \put(10,10){\vector(-1,-1){10}}
20    \put(15,0){\line(1,2){10}}
21    \put(15,10){\line(1,1){10}}
22    \put(15,20){\line(1,0){10}}
23    \put(30,0){\framebox(10,20){}}
24 \end{picture}%
```

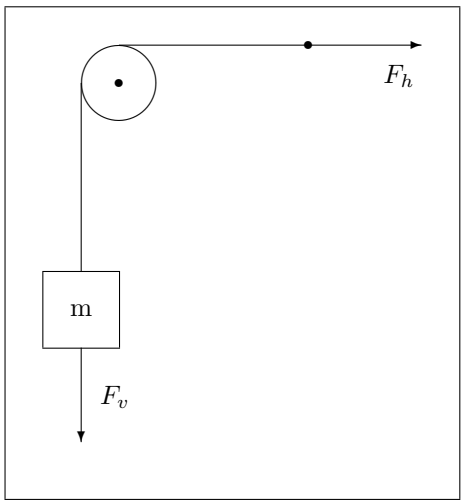


FIGURA 18: Questa figura è composta da cinque oggetti: \circle*, \circle, \framebox, \vector, \line

Il codice della figura 18 contiene un comando nuovo alla riga 2, \frame, che come argomento racchiude l'intero ambiente picture. Il suo unico scopo è quello d'inserire la cornice alla figura. La prima riga contiene l'indicazione dell'unità di misura e a questo proposito si ricorda che, modificando tale unità, la figura risulterebbe alterata in rapporto all'entità della modifica apportata e solo il testo alle righe 4, 10, 11 non ne verrebbe coinvolto. Se si desidera ingrandire o rimpicciolire il disegno è quindi consigliabile usare un'altra strada, che non sia quella di scalare l'unità di misura. Una buona *lente* è data dal comando \scalebox{sca}{oggetto} del pacchetto graphics, dove sca è un numero che indica il fattore d'ingrandimento. Tale estensione si può richiamare inserendo \usepackage{graphics} nel preambolo del documento.

Codice figura 18

```
\begin{center}\setlength{\unitlength}{0.5cm}
 \frame{\begin{picture}(12,13)(0,0)
 \put(2,4){\vector(0,-1){2.5}}
 \put(1,4){\framebox(2,2){m}}
 \put(2,6){\line(0,1){5}}
 \put(3,11){\circle{2}}
 \put(3,12){\vector(1,0){8}}
 \put(3,11){\circle*{0.2}}
 \put(8,12){\circle*{0.2}}
 \put(2.5,2.5){$F_v$}
 \put(10,11){$F_h$}
\end{picture}}%
```

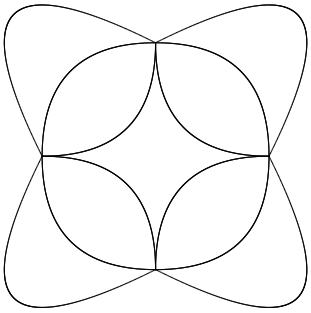


FIGURA 19: Un disegno con 12 curve `\qbezier`

La figura 19 è ottenuta con il seguente codice:

Codice per la figura 19

```
\begin{center}
\setlength{\unitlength}{1mm}
 \begin{picture}(60,60)(0,0)
  \qbezier(30,15)(15,15)(15,30)
  \qbezier(15,30)(15,45)(30,45)
  \qbezier(30,45)(45,45)(45,30)
  \qbezier(45,30)(45,15)(30,15)
  \qbezier(30,15)(0,0)(15,30)
  \qbezier(15,30)(0,60)(30,45)
  \qbezier(30,45)(60,60)(45,30)
  \qbezier(45,30)(60,0)(30,15)
  \qbezier(30,15)(30,30)(15,30)
  \qbezier(15,30)(30,30)(30,45)
  \qbezier(30,45)(30,30)(45,30)
  \qbezier(45,30)(30,30)(30,15)
  \end{picture}%
 \end{center}
```

La costruzione della figura 19 è semplice: le curve si saldano in corrispondenza del primo e terzo punto di ciascuna dichiarazione; se si prova a modificare il secondo punto di ciascuna curva, il risultato sarà una maggiore o minore concavità della curva. Non è semplice immaginare il risultato finale, occorre fare qualche prova compilando più volte il codice. Onde evitare di perdere tempo in tentativi inutili, prima di ottenere quanto si desidera, è bene evitare di seguire la tentazione che ci porta a scrivere immediatamente il codice del disegno. È preferibile disegnare con cura tutti gli elementi della figura su un foglio quadrettato o di carta millimetrata e aiutarsi con l'analoga griglia elettronica messa a disposizione dal pacchetto `graphpap`. Tale estensione deve essere richiamata nel preambolo attaverso `\usepackage{graphpap}`. Nella figura 20 è riportato un esempio di griglia. La riga 24, che riporta il comando `\graphpaper`, costruisce la griglia con le stesse dimensioni dichiarate per l'ambiente e con punto di riferimento l'origine del sistema di coordinate. Le righe della griglia sono spaziate di due unità.

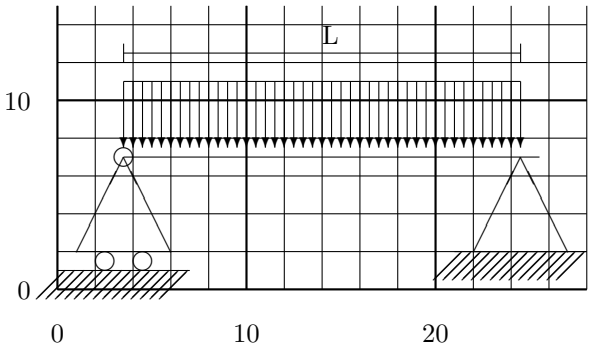


FIGURA 20: Disegno con griglia di riferimento.

Codice per la figura 20

```
\begin{center}
\setlength{\unitlength}{0.25cm}
\begin{picture}(28,15)(0,0)
  \put(0,1){\line(1,0){7}}
  \multiput(0.4,1)(0.5,0){14}{%
    \line(-1,-1){1.5}}
  \put(2.5,1.5){\circle{1}}
  \put(4.5,1.5){\circle{1}}
  \put(1,2){\line(1,0){5}}
  \put(1,2){\line(1,2){2.5}}
  \put(6,2){\line(-1,2){2.5}}
  \put(3.5,7){\circle{1}}
  \put(3.5,7){\line(1,0){22}}
  \put(22,2){\line(1,0){5}}
  \put(22,2){\line(1,2){2.5}}
  \put(27,2){\line(-1,2){2.5}}
  \put(21,2){\line(1,0){7}}
  \multiput(21.4,2)(0.5,0){14}{%
    \line(-1,-1){1.5}}
  \multiput(3.5,11)(0.5,0){43}{%
    \vector(0,-1){3.5}}
  \put(3.5,11){\line(1,0){21}}
  \put(3.5,12.5){\line(1,0){21}}
  \put(3.5,12){\line(0,1){1}}
  \put(24.5,12){\line(0,1){1}}
  \put(14,13){L}
  \graphpaper[2](0,0)(28,15)
\end{picture}
```

10 Conclusione

Come si è visto, l'ambiente offre un metodo veloce per realizzare disegni semplici e man mano che i disegni diventano più complessi il codice si

'appesantisce' di pari passo. Per quanto l'ambiente picture possa dare le sue soddisfazioni, non è lo strumento più adatto per realizzare disegni complessi; pacchetti quali epic, eepic, xy-pic, pstricks offrono qualche strumento in più rispetto a picture. Esistono poi numerosi altri pacchetti grafici per quasi ogni necessità e la cui documentazione è facilmente reperibile in rete.

Riferimenti bibliografici

Helmut Kopka, P. W. D. (2004). *Guide to LaTeX*. Addison-Wesley, Reading, MA, USA, 4ª edizione.

Lamport, L. (1994). *LaTeX: A Document Preparation System: User's Guide and Reference Manual*. Addison-Wesley, Reading, MA, USA, 2ª edizione. Aggiornato per LaTeX 2ε.

▷ Massimo Caschili
Milano
massimo.caschili@tiscali.it

Norme tipografiche per l'italiano in LaTeX

*G. Cevolani**

Sommario

Il presente articolo descrive sinteticamente le principali norme tipografiche della lingua italiana, da seguire nella composizione di articoli, tesi o libri di carattere generale. Per ogni regola discussa, si mostra come applicarla in LaTeX; tuttavia, l'articolo può risultare utile agli utenti di qualsiasi altro programma di composizione del testo.

1 Introduzione

Il presente articolo vuole offrire una sintesi delle principali "norme" tipografiche italiane, utili nella composizione di un normale testo quale un articolo, un libro o una tesi di carattere generale.

La parola "norme" va qui presa in senso piuttosto ampio: in italiano, come in tutte le altre lingue, non esistono che pochissime regole tipografiche realmente universali e vincolanti, mentre molti aspetti del testo dipendono piuttosto da convenzioni e abitudini o dal gusto dell'autore o dell'editore del testo. Per questo motivo, il presente documento non ha la pretesa di fornire un *modello* generale di scrittura ma semplicemente di riassumere le convenzioni più comunemente seguite nella composizione tipografica di un testo in italiano.

Inoltre, nel sèguito discuteremo soltanto le norme tipografiche più generali, ignorando questioni relative alla composizione di documenti in àmbito tecnico, scientifico o specialistico. Testi di questo genere seguono le norme proprie della disciplina in questione, che sono differenti da quelle generiche qui discusse e spesso le approfondiscono.

L'italiano in LaTeX

Questo articolo è stato scritto pensando alla composizione di un testo al calcolatore col programma di tipografia digitale LaTeX. Dopo l'introduzione di ogni regola, si ricorda quindi al lettore come applicarla utilizzando i comandi LaTeX adatti. Ovviamente, le norme in quanto tali sono del tutto indipendenti dal programma usato per comporre il testo, e possono quindi venir lette dagli utenti di qualsiasi altro programma di scrittura.

Per quanto riguarda LaTeX, conviene qui anticipare alcuni argomenti che verranno trattati nel sèguito.

*Desidero ringraziare Cesare Cevolani e un recensore anonimo per aver letto e commentato con grande attenzione una precedente versione di questo articolo, e gli utenti del *forum* sulla tipografia del Gruppo Utilizzatori Italiani di TeX (`www.guit.ssup.it/forum`) per le domande, i suggerimenti e le discussioni sui temi qui trattati.

Innanzi tutto, qualsiasi documento LaTeX in italiano dovrebbe richiamare nel preambolo il pacchetto babel con l'opzione italian (descritto nel § 5.1 e in Braams (2005, § 24)), che attiva la sillabazione per la lingua italiana e imposta correttamente tutte le stringhe predefinite (per esempio il comando `\chapter` produce "Capitolo" invece di "Chapter"). Il pacchetto babel offre anche alcune facilitazioni per scrivere le virgolette alte e basse (§ 3.3), le unità di misura, gli apici e i pedici (§ 5.8).

A questo proposito, va notato che dalla tastiera italiana mancano alcuni caratteri (come il simbolo per l'accento grave e la tilde, per esempio) utili per introdurre alcuni comandi LaTeX. In questi casi, occorre inserire questi caratteri nel sorgente personalizzando la tastiera o richiamandoli col corrispondente codice ascii (il modo per farlo dipende dal sistema operativo).

Infine, due altri pacchetti standard sono fortemente consigliabili. Il primo è il pacchetto inputenc che permette di inserire direttamente i caratteri presenti sulla tastiera, come le lettere accentate, che altrimenti non sono disponibili (§ 2.1). Occorre specificare la codifica adatta al sistema operativo usato, come riportato per esempio in Trettin (2004, § 2.2.6).

Il secondo (che non riguarda in particolare l'italiano) è il pacchetto fontenc, che permette di selezionare i caratteri da usare nel documento; la codifica standard da usare in LaTeX si chiama "T1". Il preambolo di un documento in italiano dovrebbe quindi contenere almeno le seguenti righe:

```
\documentclass{article}
...
\usepackage[italian]{babel}
\usepackage[<codifica>]{inputenc}
\usepackage[T1]{fontenc}
...
\begin{document}
```

2 Accento e apostrofo

L'italiano ha tre tipi di accento che possono venir segnalati graficamente: il tonico, il fonico e il circonflesso.

2.1 Accento tonico e accento fonico

Il primo tipo è l'accento *tonico* che segnala, in una parola, la vocale su cui appunto «cade l'accento». Per esempio, l'accento tonico permette di distinguere "àmbito" (di ricerca) da "ambìto" (sinonimo di "desiderato") o "pàssero" (uccello) da "passerò" (verbo).

Vocale	*Esempi*	*Codice LaTeX*
à	àmbito, città	`\‘ambito, citt\‘a`
è	è, cioè, caffè, pèsca	`\‘e, cio\‘e, caff\‘e`
é	né, perché, ché, pésca	`n\’e, perch\’e, ch\’e, p\’esca`
ì	ambìto, così!	`amb\‘\i␣to, cos\‘\i!`
ò	però, sarò, còrso	`per\‘o, sar\‘o, c\‘orso`
ó	córso	`c\’orso`
ù	più	`pi\‘u`

Tabella 1: Vocali accentate italiane e codice LaTeX corrispondente.

Il secondo tipo è l'accento *fonico*, che segnala la pronuncia *aperta* o *chiusa* di una vocale. Per esempio, l'accento fonico permette di distinguere "pésca" (in mare) da "pèsca" (il frutto), o "bótte" (di vino) da "bòtte" (nel senso di percosse). Notare che, per la fonetica italiana, solo la "e" e la "o" possono venire pronunciate aperte o chiuse (e quindi hanno l'accento), mentre la "a", la "i" e la "u" ammettono una sola pronuncia.[1]

Per distinguere questi due casi, l'accento fonico assume due forme diverse, chiamate rispettivamente *accento grave* (da sinistra in alto a destra in basso: `), che segnala un suono aperto della vocale, e *accento acuto* (da sinistra in basso a destra in alto: ´), che segnala un suono chiuso.

Forma	*Codice LaTeX*	*Codice* ASCII
Acuto ´	`\’`	39
Grave `	`\‘`	352

Benché i due tipi di accento, tonico e fonico, abbiano una funzione del tutto distinta, lo stesso segno grafico, cioè *la forma grave dell'accento fonico*, viene usato anche per segnalare l'accento tonico. L'accento tonico è quindi sempre indicato con un accento grave, ad eccezione di quei casi in cui entrambi gli accenti, tonico e fonico (chiuso), cadono sulla stessa vocale: per esempio, in "perché" e nelle altre parole «tronche».[2]

Date le cinque vocali e le due pronunce (aperta o chiusa) di "e" e di "o", esistono in italiano sette casi di vocali accentate (vedi tabella 1). Si noti che, al contrario delle vecchie macchine da scrivere o di altri programmi di composizione del testo (come *Word*), LaTeX permette di accentare senza problemi le lettere maiuscole (il codice `\‘{E}` produce direttamente «È»), che spesso vengono erroneamente rese con un apostrofo («E'», che significa "ei, egli"). In effetti, LaTeX permette di accentare *qualsiasi* lettera o carattere: per esempio `\’m` o `\‘(` producono rispettivamente "ḿ" e "̀("!

Naturalmente, è possibile definire nuovi comandi per inserire più comodamente le lettere accentate, sull'esempio di:

```
\newcommand{\ah}{\‘{a}}
```

nel qual caso occorre ricordarsi di forzare lo spazio in fine di parola: "sarà vero?" si ottiene col codice `sar\ah\␣vero?`.

Alternativamente è possibile usare il pacchetto `inputenc` per inserire i caratteri accentati direttamente da tastiera; ad esempio

```
\usepackage[ansinew]{inputenc}
```

usa la codifica per l'ambiente *Windows*. Si noti tuttavia che per alcuni caratteri, ad esempio la "o" chiusa (ó), assenti dalla tastiera italiana, occorrerà comunque usare il codice esplicito.

2.2 Accento circonflesso

Il terzo tipo di accento è l'accento circonflesso, ^, che viene usato a volte, e comunque non obbligatoriamente, per distinguere parole omografe, soprattutto nel caso dei plurali in -ii: per esempio, distingue *odî* (plurale di *odio*) da *odi* (voce del verbo udire), *desiderî* da *desìderi*, *varî* da *vari*, ecc.

In LaTeX, si introduce col comando `\^` posto davanti alla lettera da accentare, per esempio: `var\^\i`.

2.3 Apostrofo

L'*apostrofo*, ', segnala normalmente la caduta della parte finale di una parola, come in: *un'altra*, *un po'*, *da'* (imperativo), ecc. Questa regola non è tuttavia fissa e sempre rispettata; fra le eccezioni (in cui l'uso prevale sulla regola) ci sono ad esempio *piè* (e non *pie'*) e *fé* (sia nel senso di "fede" che di "egli fece"), anche se in questo secondo caso l'uso non è costante (Serianni, 1988, pp. 48, 68–69). Occorre quindi, caso per caso, consultare un dizionario.

A quanto pare (Serianni, 1988, p. 45) è lecito andare a capo subito dopo l'apostrofo anche se LaTeX tende sempre a evitarlo. In ogni caso sembra preferibile cercare altre soluzioni, cosa che LaTeX fa automaticamente.[3]

Fra gli usi particolari, l'apostrofo indica una riduzione delle cifre di un anno (per esempio: «il '68») ed è usato come simbolo per i minuti primi, di angolo e di tempo (per esempio: «ho impiegato

1. Si provi per esempio a pronunciare una "a" chiusa.
2. Si noti che il termine "accento" viene quindi usato in almeno tre sensi: per l'intonazione (fenomeno tonico), per la pronuncia della vocale (fenomeno fonico) e per il simbolo grafico impiegato per segnalare entrambi i primi due.
3. Ammesso naturalmente che sia stata attivata la sillabazione italiana come descritto nel § 5.1.

20' a leggere questa guida»). Si noti che in contesti matematici LaTeX cambia correttamente la resa grafica dell'apostrofo, come in P' (codice: `$P'$`), che risulta quindi equivalente al comando `\prime`: P' (codice: `$P^\prime$`).

2.4 Uso di accenti e apostrofi

Molto spesso, nella scrittura quotidiana, sia l'accento tonico sia l'accento fonico *non* vengono segnalati graficamente, lasciando al contesto il compito di distinguere fra parole omografe (si trovano quindi normalmente "Il mio ambito di ricerca" e "Il premio è molto ambito", o "Alla domenica vado a pesca" e "Ho mangiato una buona pesca", senza accento alcuno).

L'unico caso di uso obbligatorio è quello dell'accento *tonico* sulle parole *tronche*, come "perché", "cioè", ecc.

In generale, è comunque consigliabile segnalare un accento di qualsiasi tipo (tonico, fonico e circonflesso) sulla versione meno usuale di parole omografe (per esempio «àncora» e «ancora», quest'ultima senza accento perché di uso comune). Tuttavia, come per le parole straniere (§ 5.6), che una parola sia o meno «di uso comune» dipende essenzialmente dal destinatario dello scritto: quindi in un testo di matematica (per esempio) si potrà scrivere «funzione monotona» al posto di «monotòna» (o, per chiarezza, usare l'accento solo alla prima occorrenza della parola).

L'apostrofo si usa obbligatoriamente nei casi indicati sopra.

3 Punteggiatura e spaziatura

La punteggiatura italiana comprende i segni di interpunzione, le parentesi, le virgolette, i puntini di sospensione, i trattini e altri simboli come asterisco e sbarretta.

Per quanto riguarda la spaziatura in LaTeX, si tenga innanzi tutto presente che esistono, oltre a quello normale, due ulteriori tipi di spazio: lo *spazio sottile* (comando: `\thinspace`) e lo *spazio insecabile* (comando: `~`, cioè *tilde* nel sorgente). Il primo serve appunto a inserire uno spazio più sottile del normale, e se ne vedrà qualche esempio più avanti.

Il secondo serve per inserire uno spazio che non può essere spezzato dalla sillabazione. È utile in varie occasioni, per esempio quando di scrive «Ing. Rossi» o «M. Rossi» ed è bene non separare le due parti dell'espressione: si scrive allora «`Ing.~Rossi`» e «`M.~Rossi`» (la tilde corrisponde al codice ASCII 126). Stessa cosa per quanto riguarda i riferimenti: per esempio conviene scrivere «`si veda la figura~\ref{fig}`» per impedire che il numero della figura in questione e la parola "figura" vengano separati.

Nello stile anglosassone, lo spazio dopo il punto finale di un periodo è maggiore del normale spazio fra parole. L'uso continentale non segue invece questa regola. La tilde è utile anche dopo le abbreviazioni, in modo che LaTeX non lasci lo spazio aggiuntivo dopo il punto dell'abbreviazione stessa interpretandolo come punto fermo (di fine periodo). Il comando

```
\frenchspacing
```

nel preambolo elimina lo spazio aggiuntivo dopo qualsiasi punto in tutto il documento.

3.1 Punteggiatura e spazi

Esistono alcune regole fisse relative all'uso degli spazi prima e dopo i segni di interpunzione:

Segni di interpunzione.

Tutti i *segni di interpunzione* (punto, virgola, punto e virgola, due punti, punti interrogativo ed esclamativo, ecc.) vanno attaccati alla parola che precede e separati per mezzo di uno spazio da quella che segue. Quindi in generale dopo ogni segno di interpunzione va battuto uno spazio.

Apostrofo.

L'apostrofo segue la regola precedente, ad eccezione dei casi (che sono però i più comuni) come "un'oca", "un'altra", ecc., che non richiedono uno spazio dopo l'apostrofo.

A volte l'apostrofo si "scontra" con le virgolette, per esempio in *l'"unico"*. Una soluzione possibile è naturalmente usare le virgolette basse (vedi §3.3); se però si sono scelte le virgolette alte, conviene allora inserire uno spazio sottile (`\thinspace`) fra l'apostrofo e le virgolette: il codice «`l'\thinspace ``unico''`» produce la forma, più leggibile, *l'"unico"*.

Virgolette e parentesi.

Sia le virgolette che le parentesi vanno attaccate alle parole che racchiudono, e separate con uno spazio, sia prima sia dopo, dal resto della frase (a meno che non siano a fine frase, come qui). Quando un segno d'interpunzione ricorre dentro a virgolette o a parentesi, la frase o il periodo finirà, come questo, con un punto *fuori* dalla parentesi stessa (ovviamente!). Si noti che solo punto interrogativo e punto esclamativo stanno di norma dentro alle parentesi (Serianni, 1988, p. 67). (Si veda il §3.2 per la possibilità di inserire un'intera frase o periodo fra parentesi, in modo tale che, come in questo caso, il punto finisca *dentro* la parentesi.)

Puntini di sospensione.

I *puntini di sospensione* sono sempre e solo tre, e, come gli altri segni di interpunzione, sono attaccati alla parola che precede e separati con uno spazio da quella che segue... in questo modo (Serianni, 1988). Se usati per indicare un'*omissione* in una citazione «è bene [...] inserire i puntini entro parentesi quadre o tonde» (Serianni, 1988, p. 64), come in questo caso.

Il comando LaTeX da usare è `\dots` o `\textellipsis` (mai inserire a mano tre punti separati), eventualmente forzando uno spazio, `\dots\`, a meno che non sia a fine frase. Si veda tuttavia il § 5.5 per il corretto uso di questi comandi.

Trattini.

Per il *trattino* lungo, o lineetta (vedi § 3.4), sembra consigliabile usare la spaziatura doppia, prima e dopo — come in questo caso.

Il trattino breve, invece, non richiede alcuno spazio, né quando separa cifre, come in "pp. 23-24", né quando separa parole, come in "guerra-lampo".

3.2 Parentesi

Le parentesi normalmente utilizzate in italiano sono le parentesi *tonde* — ( ) — e le parentesi *quadre* — [ ]. Le parentesi graffe — { } — e uncinate — ⟨ ⟩ (codice LaTeX: `$\langle$ $\rangle$`) o < > (da tastiera col pacchetto fontenc) — vengono usate solo in ambiti tecnici, tipicamente in matematica.

L'uso normale delle parentesi è quello di inserire un *inciso* nel discorso (cioè una frase relativa a quella principale ma non strettamente necessaria, dal punto di vista logico, al discorso stesso, come in questo caso).

Le parentesi quadre vengono usate quasi solo in due casi: come parentesi interne a parentesi (come [anche se non mi sembra molto bello] in questo caso) e nelle citazioni per indicare «un commento dello scrivente [come in questo caso]» (Serianni, 1988, p. 67) in modo che non sia confuso con parole dell'autore. In quest'ultimo caso rientra anche quello dell'omissione volontaria (che è comunque un commento), segnalata con [...] (§ 5.5).

Tutti i segni di interpunzione, eccetto i punti esclamativo e interrogativo, vanno posti *dopo* la parentesi chiusa (vedi § 3.1).

In italiano sembra poco diffuso — ma comunque lecito (Lesina, 1987, pp. 94-95) — l'uso di racchiudere un intero periodo all'interno delle parentesi, abitudine invece piuttosto comune per esempio in inglese. (In questo caso, la punteggiatura viene anch'essa compresa fra parentesi, come qui.)

3.3 Virgolette

In italiano, esistono tre tipi di virgolette: le *basse* (« », dette anche «francesi», «caporali» o «sergenti»), le *alte* (“ ”, dette anche «inglesi») e gli *apici* (‘ ’). La forma *alto-basso* (“ „) non è usata in tipografia ma solo nella scrittura a mano (Serianni, 1988, p. 64).

Uso delle virgolette

In generale, le virgolette servono a "staccare" graficamente una parola o un'espressione dal resto del testo. Per esempio, sono usate per riportare parole altrui (come nel caso del *discorso diretto*) o per segnalare che una certa parola o frase è usata in un senso speciale differente da quello ordinario. Alcuni usi tipici delle virgolette sono i seguenti:

- Citazioni (e "intercitazioni", cioè citazioni dentro a citazioni) di pensieri, discorsi e scritti altrui;
- Menzione (invece che uso) di una espressione, come in: *"cane" ha quattro lettere*, dove la parola "cane" non viene usata ma solo menzionata;
- Significato traslato o speciale di un'espressione, per evidenziarne l'uso ironico, dispregiativo o semplicemente scherzoso, come in *Questi grandi "esperti" dicono...* o in *È un "mago" del computer*;
- Parole inusuali o straniere, come in *Il "guru" locale di LaTeX*;
- Titoli di opere.

Non esistono regole fisse o comunemente accettate per l'uso dei vari tipi di virgolette nei vari casi sopra elencati (e negli altri tanti possibili). Inoltre, in quasi tutti questi casi l'uso del *corsivo* (§ 4) concorre con quello delle virgolette.

La cosa fondamentale, quindi, è fare una scelta iniziale ragionevole ed attenersi coerentemente ad essa nel resto del documento. Per quanto riguarda il presente documento, le principali scelte di stile, comprese quelle relative alle virgolette, sono riportate nella tabella 5 a pagina 41.

Comandi LaTeX

Alcuni dei comandi LaTeX per gestire le virgolette sono riassunti in tabella 2.

Il pacchetto babel con opzione italian (§ 5.1) rende attivo il carattere `"` (virgolette alte da tastiera) e definisce i due comandi `"<` e `">` rispettivamente per le virgolette basse aperte e chiuse (Braams, 2005, § 24), che fra l'altro gestiscono automaticamente la spaziatura.

Un secondo metodo è utilizzare il pacchetto inputenc (con codifica appopriata, ad esempio ansinew per l'ambiente Windows) e inserire le virgolette basse direttamente nel sorgente, magari con un comando definito come

```
\newcommand{\caporali}[1]{«#1»}.
```

Per quanto riguarda le virgolette alte, si possono inserire col codice `‘‘ ’’`, che però risulta scomodo dato che l'accento grave ‘ manca dalla tastiera italiana,[4] o, sempre utilizzando il pacchetto babel con opzione italian, col codice `"" ’’`.

Infine, gli apici si introducono col codice `‘ ’`.

La tabella 2 riassume i comandi LaTeX per le virgolette e i pacchetti necessari per introdurli.

4. Alcuni *editor*, come TEXnicCenter, rendono possibile sostituire automaticamente le normali virgolette da tastiera in virgolette LaTeX.

Virgolette		*Codice LaTeX*	*Codice* ASCII	*Pacchetto*
Basse	«testo»	`"< testo ">`	[tastiera]	babel (opzione italian)
		`«testo»`	`174,175`	*idem* + inputenc
Alte	“testo”	`""testo''`	[tastiera]	babel (opzione italian)
		`‘‘testo’’`	`352,39`	—
Apici	‘testo’	`‘testo’`	`352,39`	—

Tabella 2: I tre tipi di virgolette e i relativi comandi LaTeX.

Si noti, nella prima riga, che gli spazi vengono automaticamente eliminati all'interno dei caporali, come vuole la tipografia italiana (ma non quella francese).

Risulta utile, a mio parere, definire tre nuovi comandi per gestire le virgolette. Nel preambolo del presente documento, ad esempio, si trovano le seguenti definizioni (i nomi sono completamente arbitrari):

```
\newcommand{\cita}[1]{«#1»}
\newcommand{\cit}[1]{‘‘#1’’}
\newcommand{\citi}[1]{‘#1’}.
```

Questo permette di separare il ruolo logico delle virgolette (citazione, significato metaforico, ecc.) dal segno grafico utilizzato per indicarlo; se ad esempio volessi eliminare i caporali dal presente documento, per utilizzare la virgolettatura inglese, mi basterebbe ridefinire `\cita` allo stesso modo di `\cit`.[5]

3.4 Trattino (o lineetta)

Il trattino (o «trattina», Fioravanti (1987)) ha due lunghezze nella stampa, *breve* e *lunga*.

Quello breve si usa per separare parole (per esempio nei composti) o cifre (come nell'indicazione delle pagine di una citazione).

Quello lungo può introdurre un discorso diretto (normalmente usandolo solo in apertura, dopo i due punti; vedi § 5.7) o per racchiudere — come in questo caso — un inciso. L'uso del tratto lungo per introdurre il discorso diretto non è comune, dato che spesso vengono utilizzate le virgolette (vedi § 3.3 e § 5.7).

Trattino		*LaTeX*	*Uso*
Breve	-	`-`	a capo, fra cifre, composti.
Lungo	—	`---`	discorso diretto, inciso.
“Meno”	–	`$-$`	matematica.

In italiano non sembra invece esistere il trattino “medio”, – (codice LaTeX: `--`), usato in inglese per separare numeri o cifre (Lamport, 1994, pp. 14, 170).[6]

Un quarto tipo di trattino può essere considerato il segno “meno” dell'operazione aritmetica, che si ottiene con un normale trattino breve in ambiente matematico (`$-$`).

3.5 Asterisco e sbarretta

La *sbarretta*, /, si usa, senza spazi né prima né dopo, per indicare un'*alternativa* tra due possibilità, come nel tipico “e/o”. Quindi non dovrebbe essere usata per indicare giustapposizione o composti, per cui si usa il trattino (§ 3.4). Per esempio: “i passeggeri diretti a Torino/Milano” (cioè a Torino o a Milano) ma “la linea Torino-Milano”.[7]

La sbarretta si utilizza anche per scrivere le frazioni numeriche, come 3/4: si veda tuttavia il § 5.8 per la corretta scrittura di quantità numeriche. Non si confonda ovviamente la sbarretta (*slash*: /), che può essere inserita direttamente da tastiera, con il *backslash* (\) riservato ai comandi LaTeX.

L'*asterisco*, *, si usa praticamente solo in tre occasioni (Serianni, 1988, pp. 67-68): come simbolo della nota a piè di pagina, in numero di tre per indicare omissione volontaria (“nel paese di * * *”, codice: `\mbox{*\thinspace*\thinspace*}`[8]) e in linguistica per indicare forme non attestate, scorrette o inaccettabili (**che io vadi*).

4 Stile del carattere

I caratteri utilizzati in italiano oltre a quello normale del testo, a volte chiamati *font* dall'inglese, comprendono lo stile *corsivo*, **grassetto** e maiuscoletto. Non sembra utilizzato invece lo stile *slanted* (“inclinato”).

Altri stili possono essere utilizzati (con parsimonia) per esigenze particolari: per esempio, nel presente documento si usa il carattere tipo `macchina da scrivere` per evidenziare i comandi LaTeX, e

5. In generale, occorre trovare un compromesso fra la comodità e il numero di nuovi comandi definiti dall'utente (che, se esagerato, rischia di rendere il sorgente “illeggibile”); mi sembra che, in questo caso, la comodità superi il rischio.

6. (Meynet, 2000, p. 31) tuttavia chiama “trattino di divisione” il trattino breve, e “lineato”, rispettivamente “breve” e “lungo”, il trattino medio e il tratto lungo. Utilizza il lineato medio come segno di apertura di un paragrafo e come segno di divisione per i nomi degli autori.

7. Un uso eccezionale della sbarretta si trova negli indirizzi postali, con «c/o» (che sarebbe l'abbreviazione dell'inglese *care of*, equivalente al nostro “all'attenzione di”).

8. Meglio, a mio parere, usare lo spazio *sottile* che lo spazio normale. A volte si trovano anche i tre asterischi non spaziati.

Stile	*Locale*	*Globale*
Corsivo	`\textit{<testo>}`	`{\itshape <testo>}`
Grassetto	`\textbf{<testo>}`	`{\bfseries <testo>}`
MAIUSCOLETTO	`\textsc{<testo>}`	`{\scshape <testo>}`
Enfatizzato	`\emph{<testo>}`	`{\em <testo>}`

TABELLA 3: I comandi LaTeX per gestire lo stile del carattere.

il carattere senza grazie per indicare i "pacchetti" LaTeX.

I comandi LaTeX per gestirli, nella versione locale e in quella globale, sono indicati in tabella 3 (TRETTIN, 2004, p. 9).

Lo stile "evidenziato" o "enfatizzato" è reso normalmente col corsivo ma riveste un ruolo logico differente, discusso nel § 4.4.

4.1 Corsivo

Il corsivo si usa nel testo principalmente:

- per enfatizzare le parole *importanti* o comunque da evidenziare rispetto al resto del testo (vedi § 4.4);
- per le *parole straniere* (anche latine) non entrate nell'uso italiano (vedi § 5.6), ad esempio *guillemet* o *politically correct*;
- per i *titoli* di opere citate, ad esempio *La Divina Commedia* o *The TEXbook*;

In bibliografia, si usa per i titoli dei libri e delle collezioni, ma questo dipende dallo stile bibliografico utilizzato (vedi § 5.10).

4.2 Grassetto

Il grassetto si usa quasi esclusivamente per i titoli dei paragrafi, capitoli e delle altre suddivisioni del testo, come nel presente documento. Si tende di norma a non usarlo per enfatizzare le parole, uso per cui esiste già il corsivo, o comunque a utilizzarlo con moderazione, per non appesantire l'aspetto della pagina.

LaTeX usa automaticamente il grassetto per i titoli di tutti i sezionamenti (`\chapter`, `\section`, `\paragraph`, ecc.) ma non per il titolo generale del documento (`\title`, `\author` e `\date`), che occorre sistemare a mano (per esempio: `\title{\bfseries <titolo>}`) o ridefinendo il comando relativo.

4.3 Maiuscoletto

Il maiuscoletto viene usato in italiano, con una certa regolarità, quasi esclusivamente per i nomi degli autori citati in bibliografia, come in questo caso: LAMPORT (1994). Questa convenzione, tuttavia, dipende dallo stile bibliografico utilizzato (vedi § 5.10).

4.4 Corsivo *vs* enfatizzato

Spesso, come nel presente documento, il *corsivo* viene utilizzato per evidenziare parole importanti che vanno messe in risalto. Tuttavia, lo stesso scopo può essere talvolta raggiunto in altri modi, per esempio con una sottolineatura, a discrezione dell'autore e dello stile di pubblicazione.

Per questo motivo, è importante tenere separati i due ruoli logici del corsivo e dell'enfasi. A tal fine, LaTeX mette a disposizione due comandi distinti, `\textit` (e `\itshape`) per il corsivo (che è un particolare stile di carattere) e `\emph` (e `\em`, da "emphasize") per indicare l'enfasi. La differenza si nota nell'esempio riportato in figura 1.

Utilizzare il comando appropriato nei due diversi casi è importante perché, se si decidesse a un certo punto di cambiare lo stile dell'enfasi (per esempio con il pacchetto ulem), il comando `\emph` verrebbe di conseguenza ridefinito, mentre `\textit` rimarrebbe (ovviamente) inalterato. Si noti, in tabella 2, la differenza di comportamento nei due casi, il primo sbagliato e il secondo corretto, qualora si cambi lo stile dell'enfasi (il pacchetto ulem fa sì che `\emph` sottolinei la parola invece di metterla in corsivo).

Usare `\emph` per scrivere in corsivo o, viceversa (errore meno comune), usare `\textit` per enfatizzare una parola può quindi portare a effetti indesiderati.

5 Composizione del testo

5.1 Sillabazione

In LaTeX, la sillabazione automatica viene attivata utilizzando il pacchetto babel con l'opzione italian.[9] Il pacchetto può venir richiamato normalmente nel preambolo del documento:

```
\usepackage[italian]{babel},
```

o globalmente come opzione di classe, per esempio:

```
\documentclass[italian]{article},
```

in modo da venir riconosciuto automaticamente anche da eventuali altri pacchetti. In questo modo, risultano anche definiti alcuni nuovi comandi e

9. Perché la sillabazione venga effettivamente attivata occorre controllare che le definizioni per la lingua italiana siano caricate: il procedimento per farlo dipende dalla distribuzione LaTeX. In una distribuzione MiKTeX per *Windows*, per esempio, si attiva la lingua italiana selezionando la relativa casella nella scheda "Languages" delle *MiKTeX Options*.

```
Questa è una \emph{parola}
enfatizzata in una frase normale
```
Questa è una *parola* enfatizzata in una frase normale

```
\textit{Questa è una \emph{parola}
enfatizzata in una frase in corsivo}
```
Questa è una parola *enfatizzata in una frase in corsivo*

FIGURA 1: La differenza fra lo stile di carattere corsivo e l'enfasi.

```
\usepackage{ulem}
```

```
Questa parola è in \emph{italics},
questa è \emph{sottolineata}.
```
Questa parola è in italics, questa è sottolineata.

```
Questa parola è in \textit{italics},
questa è \emph{sottolineata}.
```
Questa parola è in *italics*, questa è sottolineata.

FIGURA 2: La differenza fra `\textit` e `\emph` quando si cambia lo stile dell'enfasi. Il secondo caso è quello corretto.

"scorciatoie" per rendere più facile l'inserimento di virgolette (§ 3.3), apici e unità di misura (§ 5.8): si veda (BRAAMS, 2005, pp. 148 e seguenti). (Un'utile sintesi delle funzioni del pacchetto babel per l'italiano, come di molti altri pacchetti, si trova in GREGORIO (2005).)

In alcuni casi, la sillabazione automatica può spezzare in modo non corretto alcune parole. Tali parole vanno allora elencate nel preambolo all'interno del comando `\hyphenation`, separate fra loro da uno spazio e dividendo le sillabe con un trattino: per esempio `\hyphenation{Ci-ce-ro-ne an-co-ra}`. Alternativamente, è possibile segnalare i punti in cui spezzare una parola direttamente nel testo, col comando `\-` (per esempio: `Ci\-ce\-ro\-ne`), cosa utile in particolare durante la revisione finale del documento, quando si risolvono i casi di "bad box".

Se il testo contiene frasi o interi periodi in altre lingue (per esempio, un *abstract* o una citazione lunga) occorre attivare la relativa sillabazione come opzione del pacchetto babel o della classe (per esempio: `\usepackage[english,italian]{babel}`). In questo caso, la lingua principale è quella indicata per ultima fra le opzioni; per introdurre porzioni di testo in una delle altre lingue specificate si usano i comandi `\selectlanguage`, `\foreignlanguage` o l'ambiente otherlanguage.

Per esempio, `\selectlanguage{english}` attiva la lingua inglese dal punto in cui viene richiamato in poi (o all'interno dell'ambiente in cui viene richiamato); `\foreignlanguage{english}{...}` scrive il testo contenuto nel secondo argomento con le regole della lingua inglese (indicata nel primo); mentre l'ambiente `\begin{otherlanguage}{english}` attiva l'inglese fino alla fine dell'ambiente stesso.

5.2 Formato della pagina

Il formato della pagina (margini, altezza e larghezza del testo, spazio per le testatine e per le note, ecc.) dipende dal tipo di pubblicazione e dalle scelte dell'editore (per esempio, le tesi di laurea impongono normalmente requisiti espliciti sul *layout*).

Il layout predefinito di LaTeX è solitamente del tutto insoddisfacente per le esigenze dell'utente medio. Essendo stato pensato per il foglio standard americano (la cosiddetta «quadrotta» o formato «letter»), produce margini troppo ampi e sproporzionati.

È del tutto sconsigliabile modificare «a mano» il formato della pagina. Invece, conviene usare pacchetti come layaureo (pensato appositamente per il foglio a4 e i formati di pagina "europei") o geometry, che permettono di modificare in modo coerente margini e dimensioni varie.

Inoltre, specialmente per lavori di àmbito umanistico, si consideri la possibilità di usare classi di documento come memoir, che fornisce un insieme unitario di comandi per gestire tutti gli aspetti della pagina e molto altro.

5.3 Paragrafi

Un paragrafo è un blocco di testo, contenente uno o più periodi, che logicamente individua una parte del discorso autonoma. Si noti che, in italiano, sia il paragrafo in questo senso sia nel senso di sotto-sezione di un capitolo (chiamata appunto `\section` in inglese) hanno lo stesso nome (per esempio, il presente paragrafo 5.3 è formato da cinque paragrafi nel primo senso).

Solitamente, la prima riga di un paragrafo viene rientrata per evidenziare lo stacco da quello precedente. In questo caso, in italiano viene rientrata anche quella del primo paragrafo di un capitolo (mentre in inglese solo il secondo paragrafo viene rientrato.) La stessa regola vale anche nel caso delle citazioni (LANZA, 1992, p. 9), quando si citi un intero paragrafo (vedi per esempio la citazione nel § 5.5). Se i paragrafi non vengono rientrati, vengono solitamente spaziati verticalmente l'uno dall'altro (LESINA, 1987, § 3.2.2); è decisamente consigliabile usare almeno uno di questi due meto-

di (rientro o spaziatura) per segnalare la fine di un paragrafo e l'inizio di uno nuovo, e facilitare così la lettura del documento.

In LaTeX, un nuovo paragrafo viene iniziato dal comando `\par` o lasciando una riga vuota nel sorgente. Non usare mai `\\` per iniziare un paragrafo: questo comando serve solo a spezzare la linea e iniziarne una nuova, senza tener conto dello spazio fra paragrafi e del rientro, come in questo caso.

Per avere il rientro anche nel paragrafo iniziale usare semplicemente il pacchetto indentfirst richiamandolo nel preambolo con:

```
\usepackage{indentfirst}.
```

Sia il rientro iniziale (`\parindent`), che lo spazio verticale fra due paragrafi successivi (`\parskip`), come del resto tutti gli altri attributi della pagina di testo, possono venire modificati agendo sui comandi LaTeX appositi (per esempio tramite `\setlength`). Modificare questi parametri senza un'adeguata considerazione del *layout* generale non è mai buona norma. Si consideri anche la possibilità di usare il pacchetto parskip e si tengano comunque presenti i consigli di (Trettin, 2004, pp. 5-6).

Vedove e orfani

In tipografia, si usa chiamare «orfano» una riga solitaria in fondo alla pagina (tipicamente la prima riga di un paragrafo) e «vedova» una riga solitaria in cima alla pagina seguente (tipicamente l'ultima riga di un paragrafo). Entrambi questi casi andrebbero evitati, facendo in modo che ci siano almeno due righe di uno stesso paragrafo sia in cima che in fondo a ogni pagina.

LaTeX è già programmato per ottenere questo effetto.[10]

5.4 Note al testo

Le *note a piè di pagina* sono normalmente numerate progressivamente e la numerazione ricomincia all'inizio di ogni capitolo.

In italiano, il numero di nota va messo sempre *dopo* all'eventuale segno di interpunzione.[11]

In LaTeX, le note si introducono col comando `\footnote{<testo>}`, che numera automaticamente le note ripartendo all'inizio di ogni capitolo. In molti libri italiani, non c'è alcuna linea orizzontale fra la note a piè di pagina ed il testo principale, linea che invece viene normalmente inserita da LaTeX. Un modo molto semplice per eliminarla è ridefinire nel preambolo il comando relativo in modo che non faccia nulla:

```
\renewcommand{\footnoterule}{}.
```

Tolta la linea, è probabile che si desideri aumentare lo spazio fra il testo e la nota; un modo per farlo è aumentare la dimensione del relativo parametro:

```
\addtolength{\skip\footins}{5mm}
```

(Si noti che i comandi in questione sono comandi TeX di basso livello: vedi (Goossens *et al.*, 1994, p. 70)).

L'uso delle note a piè di pagina (di cui comunque è bene non abusare) con la numerazione qui proposta è preferibile sia alla scelta di ricominciare la numerazione a ogni pagina sia a quella di non interrompere la numerazione a ogni capitolo. È anche la scelta di *default* in LaTeX: se si ha qualche valida ragione di cambiarla si può far uso del pacchetto footmisc, che raccoglie le funzionalità di vari altri pacchetti e permette un'ampia personalizzazione delle note.

La scelta di mettere le note a piè di pagina è la più comoda e intuitiva per il lettore, che ritrova i riferimenti e le altre informazioni nella stessa pagina. L'uso di riservare alle note un paragrafo separato alla fine del documento o del capitolo (paragrafo intitolato semplicemente *Note*) dipendeva spesso dalle difficoltà di impaginazione dovute alle note a piè di pagina (Lesina, 1987, capitolo 14), difficoltà ormai superate dagli attuali programmi di composizione e in particolare da LaTeX. Se si preferisce comunque l'elenco finale delle note si usi il pacchetto endnotes, che permette anche di avere elenchi di note separati per ogni capitolo.

5.5 Citazioni

Le *citazioni* vanno inserite fra virgolette del tipo scelto (vedi § 3.3) nel caso siano brevi (un paio di frasi al massimo) o come paragrafo separato nel caso siano lunghe.

Le *intercitazioni*, cioé le citazioni dentro un'altra citazione, vanno inserite fra virgolette di tipo diverso; nel caso di citazioni fuori corpo del testo, il problema ovviamente non sussiste. Si veda la tabella 5 per alcune scelte possibili.

Nel caso di citazioni fuori testo, il corpo del paragrafo citato è *minore* del corpo del testo: questo significa che margini, dimensioni del carattere, interlinea e tutte le altre caratteristiche del testo devono venire ridotte nelle citazioni. L'ambiente quotation del LaTeX risponde solo parzialmente a queste esigenze. Conviene dunque definire un nuovo ambiente citazione in questo modo:

```
\newenvironment{citazione}%
{\begin{quotation}%
 \small\ignorespaces}%
{\end{quotation}}
```

che permette di ottenere una citazione come la seguente (mentre il codice qui sopra è stato scritto all'interno di una `quotation`):

10. I relativi comandi, come `\clubpenalty` e `\widowpenalty`, e i possibili modi di cambiarne il comportamento, sono discussi nel *TEXbook* (Knuth, 1986).

11. Così.

> Col comando così definito è possibile racchiudere citazioni lunghe fra i comandi `\begin{citazione}` e `\end{citazione}`. `\ignorespaces` serve ad evitare uno spazio all'inizio della citazione stessa.

Se si è scelto di rientrare i paragrafi, anche nelle citazioni il periodo iniziale va rientrato (Lanza, 1992, p. 9), cosa che risulta automatica se si sta usando il pacchetto indentfirst.

Commenti all'interno di una citazione vanno inseriti fra parentesi quadre, per evitare che vengano confusi con parole dell'autore. Un'omissione volontaria nella citazione è da considerare un commento e va quindi segnalata con [...].[12] Il codice LaTeX `[\dots]` o `[\textellipsis]` in questo caso non produce una spaziatura perfetta (lascia troppo spazio dopo l'ultimo puntino). Per correggerla, utilizzare il pacchetto ellipsis, richiamandolo nel preambolo con

```
\usepackage{ellipsis},
```

che modifica automaticamente il risultato.

Un errore nel testo originale (se significativo) va segnalato nella citazione con un [*sic*] ("così!" in latino).

5.6 Parole straniere

Le parole straniere vanno in corsivo, a meno che non vengano esplicitamente «quoted» (come in questo caso) o che non siano entrate nell'uso comune. Quindi, per esempio, si scriverà: "ho visto un bel film" (comune) ma "ho mangiato un *pudding*" (non comune).

In realtà, dato che è molto difficile stabilire cosa sia entrato o meno «nell'uso comune», la regola più corretta è quella suggerita per esempio da (Fioravanti, 1987, p. 14): vanno in corsivo le parole straniere che si presumono non di uso corrente per il lettore a cui ci si rivolge. Ad esempio, in un libro di informatica "software", "computer", ecc. potranno tutte andare normalmente in tondo.

Da notare che le parole straniere usate correntemente non assumono la forma plurale: quindi "ho mangiato due *puddings*" ma "ho visto due film".

I nomi propri (come «Texas», «George», ecc.) o le denominazioni ufficiali (come «University of California», «Magna Charta», ecc.) non sono considerati parole straniere, e quindi non vanno in corsivo.

La traduzione straniera di un'espressione italiana usata nel testo può essere semplicemente messa in corsivo e fra parentesi (*bracket*), come in questo caso. Se invece l'espressione tradotta ricorre in una citazione, e si vuole indicare la forma originale, occorre inserirla fra parentesi quadre, come ogni altro commento. Per esempio: «La visione del mondo [*Weltanschauung*]...».

12. Spesso si usano i puntini semplici (Lesina, 1987, p. 195), che però sono ambigui, potendo già appartenere al testo originale.

5.7 Dialoghi

Esistono diversi modi di introdurre nel testo dialoghi o parti di essi, a seconda che siano frasi brevi e isolate o che l'intero testo sia composto da una successione di discorsi diretti. Le virgolette o il trattino lungo sono i simboli più utilizzati per indicare la fine e l'inizio di un dialogo. Alcuni esempi:

- Quando lo incontrai mi disse: «Ciao!».
- Quando lo incontrai mi disse: — Ciao!
- «Salve!», dissi io.

  «Ciao!», mi disse lui.
- Salve! — dissi io.

  Ciao! — mi disse lui.

Quando non cambia il parlante e si vuole introdurre un'interruzione si possono tenere aperte le virgolette e usare i trattini, come in un inciso: «Ciao! — dissi io — come va?».

5.8 Numeri

L'aspetto tipografico dei numeri o di espressioni numeriche è un argomento abbastanza vasto, cui è dedicato l'ottavo capitolo di Lesina (1987). Ci limitiamo qui a descrivere alcune regole generali.

Lettere o cifre?

Qualsiasi numero può venir scritto altrettanto correttamente in cifre o in lettere. La scelta dipende anche dalla natura del testo: quelli umanistici tendono a usare le lettere, quelli tecnico-scientifici le cifre. Esistono comunque alcune regole generali (Lesina, 1987, pp. 129-131):

- per numeri inferiori a 10 si tendono a usare le cifre, per numeri maggiori o uguali a 10 le lettere (in questo caso «10» è scritto in cifre perchè non indica una quantità, vedi sotto);
- se un numero, anche maggiore di 10, è all'inizio del periodo lo si scrive in lettere: «Trentatre trentini venivano da Trento...»;
- se una quantità è meramente indicativa, si usano le lettere o una forma mista: «ci saranno state mille persone», «costa almeno 2 milioni di euro»;
- se un numero non indica una quantità, come in «a pagina 5», «articolo 3», «ho preso 4 in Latino», si usano le cifre.
- se un numero è decimale (per esempio $6,5$) si usano le cifre (fanno eccezione casi come «è alto un metro e ottanta»).

Ordinali

Anche gli ordinali (primo, secondo, ecc.) si scrivono tipicamente in lettere se minori di 10 o in cifre se maggiori o uguali a 10.

Nella scrittura in cifre, l'ordinale si indica con un circoletto in posizione di apice (esponente): per esempio, "1°" che si ottiene in LaTeX col codice `1\textsuperscript{$\circ$}`, o semplicemente con `1°` se si usa il pacchetto inputenc per inserire i simboli da tastiera. (Da notare che, benché il circoletto indichi in origine la "o" finale di "primo", il simbolo relativo non è una "o" ma appunto un circolo.) Per ordinali quali "prima", "primi", "prime", ecc. sembra comunque corretto usare come invariabile la forma col circoletto (Lesina, 1987, p. 136, nota 10). Altrimenti si può usare la forma "1ª" (e relative) corrispondete al codice LaTeX: `1\textsuperscript{a}`. Il pacchetto babel con l'opzione italian permette di ottenere lo stesso identico risultato col comando `\ap` (per "apice").[13]

Quando, per indicare un ordinale, si usano i numeri romani al posto dei numeri arabi, non va usato il circoletto in esponente: per esempio, in "Carlo V" e "il XX secolo" i due numeri romani vengono letti rispettivamente "quinto" e "ventesimo" senza bisogno di altra indicazione.

Separazione fra cifre

Per numeri di cinque o più cifre, è utile inserire uno spazio fra ogni gruppo di tre cifre (partendo da destra), per evidenziare le *migliaia* e facilitare la lettura del numero stesso. Lo spazio in questione non dovrebbe essere un normale spazio (come in 1 500 000) ma invece uno *spazio sottile* (come in 1 500 000), che si ottiene col comando `\thinspace`, sia in ambiente matematico che in ambiente testo:

```
1\thinspace 500\thinspace 000.
```
[14]

È bene evitare l'uso di virgola e punto (che rappresentano rispettivamente la pratica anglosassone e quella europea) per separare le migliaia, uso che rischia di confondersi con la separazione *decimale* e di provocare gravi errori di lettura. L'uso dello spazio sottile è l'unico metodo universale corretto (Lesina, 1987, p. 131).

Nei *numeri decimali*, che si scrivono quasi sempre in cifre, il separatore fra la parte intera e quella decimale è una virgola (26,5), secondo l'uso europeo continentale (in quello anglossassone è un punto). All'interno di un testo normale, si tende a ridurre al massimo il numero delle cifre decimali: quindi si scriverà "26,5" e non "26,50". In questi casi è sempre bene inserire il numero in ambiente matematico (`$26,5$`), che corregge opportunamente la spaziatura dopo la virgola.

13. Risulta anche definito e attivo il comando `\ped` per il pedice.

14. Lo stesso effetto si ottiene, in ambiente matematico, con lo spazio matematico `\,`: si veda (Knuth, 1986, p. 5, esercizio 2.5 e relativa soluzione).

Frazioni, percentuali, unità di misura

Le *frazioni* si possono esprimere in lettere (*tre quarti*) a meno che non indichino una quantità numerica precisa. In questo caso si possono scrivere utilizzando la sbarretta, per esempio 3/4, o la forma frazionaria vera e propria, per esempio $\frac{3}{4}$ (codice LaTeX: `$\frac{3}{4}$`). Quest'ultimo uso diventa praticamente obbligatorio quando si aggiunge una frazione a un numero intero, come in $2\frac{3}{4}$. In questo caso risulta più soddisfacente (e più aderente alla pratica tipografica) usare il pacchetto xfrac che permettere di scrivere le frazioni in modo più compatto ed elegante: il codice "`1\sfrac{3}{4}`" produce $1^{3}/_{4}$.[15]

Le *percentuali* si scrivono normalmente in cifre (30%) a meno che il contesto non sia colloquiale, nel qual caso si può scrivere "30 per cento". Ricordare che il simbolo "%" è riservato in LaTeX ai commenti, e va quindi inserito con l'apposito comando "`\%`".

Le *quantità misurate* sono costituite da numeri seguiti dall'unità di misura, espressa tipicamente dal simbolo relativo. In LaTeX, è utile un comando messo a disposizione dal pacchetto babel con opzione italian, `\unit`, che permette di scrivere il simbolo dell'unità di misura, sia in ambiente testo che in ambiente matematico, nel modo corretto, cioè in tondo e con un'adeguata spaziatura (corrispondente allo spazio sottile `\thinspace`): per esempio, "20 cm" o "15 Kg".[16] Si noti che il simbolo dell'unità di misura non vuole il puntino.

5.9 Sigle

Gli *acronimi* sono espressioni formate dalle lettere o sillabe iniziali di parole componenti un'espressione che si vuole abbreviare. Esempi sono ISTAT, USA o GUIT. Gli acronimi dovrebbero essere composti tutti da maiuscole, senza spazi né puntini fra di esse (Lesina, 1987, p. 148) anche se spesso si trovano forme differenti. Quando un acronimo è ampiamente entrato nell'uso e viene pronunciato come parola, si può spesso scrivere come tale: "Fiat" e "radar" sono entrambi accettabili (Lesina, 1987, p. 149). A parte in questi casi, è sempre consigliabile, per acronimi meno noti, citarli per esteso la prima volta che li si utilizza nel testo, mettendo fra parentesi la forma estesa (per esempio: Gruppo Utilizzatori Italiani di TeX).

Le *abbreviazioni* (si veda l'appendice A) si ottengono invece dal troncamento di una parola, che viene indicata dalla sua sola parte iniziale. In casi particolari, come "sig.ra" o "Prof.ssa" comprendono anche la parte finale della parola originale. Il troncamento viene indicato con un punto: se l'abbreviazione dovesse cadere a fine periodo — caso raro, forse possibile col solo "ecc." — il punto di

15. Un altro pacchetto simile, ma reso obsoleto da xfrac, è nicefrac.

16. Si consideri anche, soprattutto in lavori scientifici, il pacchetto Slunits, interamente dedicato alla corretta scrittura delle unità di misura.

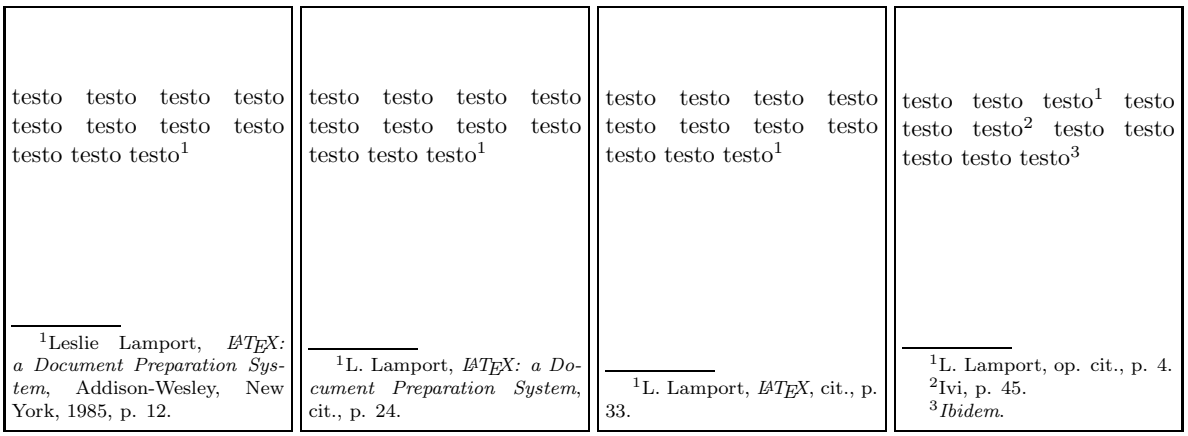


FIGURA 3: Un esempio del sistema tradizionale di riferimento bibliografico in nota a piè di pagina, coi 6 livelli di citazione possibili. Da notare che l'uso di "Ivi" richiede che la citazione immediatamente precedente sia tratta dalla stessa opera, quello di "*Ibidem*" che sia tratta anche dalla stessa pagina.

troncamento funziona anche da punto fermo (cioè, ovviamente, non si trovano due punti successivi).

Le «abbreviazioni personali», di nome e di titolo, come già notato all'inizio del §3), vanno "incollate" al nome che segue (per esempio, "M. Rossi" o "Ing. Rossi") con uno spazio insecabile (~) in modo che l'espressione complessiva non venga spezzata a fine riga.

5.10 Bibliografia

La bibliografia è forse il più tormentato fra gli aspetti tipografici della composizione di un documento. Se in generale esistono poche regole assolute per la tipografia, nel caso della bibliografia non ne esiste sostanzialmente alcuna.

Lo stile dei riferimenti bibliografici, sia nel testo sia nelle note, dipende essenzialmente dal tipo di pubblicazione, dall'editore, dai gusti del'autore e da altri fattori. Prendiamo qui in considerazione solo un esempio, abbastanza tipico di molti documenti italiani (soprattutto in ambito umanistico), rimandando a una futura occasione la trattazione esauriente della bibliografia da un punto di vista tipografico.

Lo stile qui considerato prevede l'uso del maiuscoletto sia per i nomi degli autori che dei curatori (*editors*), il corsivo per i titoli di libri o antologie (*collections*) e per i nomi delle riviste (*journals*) e le virgolette basse per i titoli degli articoli o dei contributi in antologie. Quindi un libro verrà riportato come:

> L. LAMPORT, *LaTeX: a Document Preparation System*, Addison-Wesley, New York, 1985.

e un articolo come:

> P. FLYNN, «Typographers' Inn», *TUGboat*, 25, pp. 134-136, 2004.

Vale comunque la pena ripetere che queste sono solo alcune delle tante scelte stilistiche possibili, che solitamente vengono imposte all'autore del documento dall'editore o dalla rivista per cui scrive.

Anche la scelta fra riferimenti *in nota* (a piè di pagina o in fondo al libro o articolo) o invece raccolti in un'apposita *bibliografia* dipende spesso dall'editore o dalla rivista. La prima scelta è piuttosto diffusa nelle pubblicazioni italiane, ma del tutto sconsigliabile (a mio parere) per qualsiasi pubblicazione scientifica — in senso lato, comprese le tesi di laurea, per esempio — a causa della maggior difficoltà di individuazione e lettura del riferimento. Un esempio illustrativo di questo sistema è riportato in figura 3. Il pacchetto footbib permette di inserire in nota, invece o oltre che in bibliografia, le citazioni.

La seconda scelta è quella standard in LaTeX: l'ambiente thebibliography produce appunto una sezione del documento separata contenente l'elenco dei riferimenti bibliografici. Questo ambiente non permette di personalizzare al meglio lo stile della bibliografia e delle citazioni. A questo scopo, è meglio utilizzare il programma BibTeX (magari assieme al pacchetto natbib), che separa l'aspetto stilistico della bibliografia — gestito dal file di stile `.bst` — dai riferimenti stessi — contenuti del file `.bib`. Il programma MakeBST permette infine di costruire un file di stile bibliografico personalizzato nei minimi dettagli.

6 Conclusioni e approfondimenti

Il presente articolo è una semplice sintesi delle principali norme tipografiche usate in italiano. Per un trattamento approfondito di questo tema, si veda il *Manuale di stile* di Lesina (LESINA, 1987), che rimane il principale riferimento italiano (ne esiste una nuova edizione del 1994, con una parte

Abbreviazione	*Altre forme*	*Significato*	*Codice LaTeX*
§, §§		paragrafo, paragrafi	`\S, \S\S`
(a cura di)	a c.	a cura di	
a.C.		avanti Cristo	`a.C.`
app.	ediz.	appendice	
cap., capp.		capitolo, capitoli	
cfr.	cf.	confronta	
cit.		citato	
d.C.		dopo Cristo	`d.C.`
ecc.		eccetera	
ed.	ediz.	edizione	
eq.		equazione	
es.		esempio	
et al.		*et alii* (e altri)	
fig., figg.		figura/e	
Ibidem	*Ibid.*	nello stesso passo della stessa opera della nota precedente	
(Ivi)	*Ivi*	nella stessa opera della nota precedente	
op. cit.	*Op. cit.*	opera citata	`op.␣cit.`
n°, nn.	n°, nn	numero/i	`n$^\circ$, nn.`
n.d.A.	n.d.a.	nota dell'Autore	`n.d.A.`
n.d.C.	n.d.c.	nota del Curatore	`n.d.C.`
n.d.T.	n.d.t.	nota del Traduttore	`n.d.T.`
(nota, note)	nt.; n., nn.	nota, note	
p., pp.		pagina/e	
(*passim*)		*passim*	
tab.		tabella	
ss.	sg., sgg.	seguente	
v.		verso	

Tabella 4: Le principali abbreviazioni italiane.

dedicata ai *word processors*, che non ho consultato). (Sempre della Zanichelli si può consultare il *Manuale del grafico* (Fioravanti, 1987), dedicato alla progettazione e all'impaginazione del documento.) Le *Norme redazionali* di Lanza (Lanza, 1992) sono un piccolo riassunto, comodo come riferimento rapido ma non molto completo. In Rete si possono trovare vari documenti simili al presente, a partire dalle sezione del sito dell'Accademia della Crusca (`www.accademiadellacrusca.it`, si vedano per esempio Marzullo (2002) e Setti (2002)) per arrivare a veri e propri manualetti liberamente disponibili (per esempio, (Meynet, 2000) o (Salmeri, 2001)), oltre alle tante norme redazionali messe a disposizione dalle riviste per i loro autori.

Una buona grammatica italiana (come quella di Serianni, Serianni (1988)) risolve tutti i dubbi relativi ai contenuti (accenti, apostrofi, ecc.) oltre a dare qualche indicazione utile a livello tipografico.

Per quanto riguarda LaTeX, il riferimento principale soprattutto per le questioni tipografiche è sempre il TeXbook (Knuth, 1986). Quasi tutti i manuali standard comprendono anche alcune indicazioni tipografiche generali. Esistono infine guide dedicate ad argomenti specifici, come la composizione in LaTeX della tesi di laurea: si vedano Mori (2005) e Beccari (1991).

A Abbreviazioni

La tabella 4 elenca, in ordine alfabetico rispetto alla forma abbreviata (con i simboli all'inizio), le principali abbreviazioni utilizzate in italiano. Fra parentesi si segnalano le forme che non vanno abbreviate. Se della stessa abbreviazione esistono più forme comunemente usate, esse vengono indicate; si segnala però solo il codice LaTeX corrispondente a quella principale, cioé a quella scelta dall'autore del presente documento.

Le abbreviazioni vanno usate con parsimonia, perché appesantiscono la lettura del documento, a meno che non siano molto comuni e ben note (come "p., pp." per "pagina, pagine" che sposta subito l'attenzione del lettore sul numero o numeri di pagina). L'uso del calcolatore e dei programmi di tipografia come LaTeX facilita enormemente, rispetto alle vecchie macchine da scrivere o da tipografia, la composizione e la correzione del testo, rendendo spesso inutile l'uso massiccio di abbreviazioni (introdotte per risparmiare tempo e spazio nella scrittura). Si consiglia quindi di usare le abbreviazioni solo in assenza di spazio o nel caso

Stile		*Esempi*
Enfatizzato	Corsivo (`\emph`)	È *proprio* così!
		In LaTeX, un *pacchetto* è un...
Parole straniere	Corsivo (`\textit`)	Un *editor* di testo.
Citazioni	Virgolette basse	Dico io: «LaTeX è meglio!».
		Veniva chiamato «il Re Sole».
Intercitazioni	Virgolette alte	«Mi disse: "Addio!"».
Menzione	Virgolette alte	"Cane" in inglese si dice "dog".
		"Porta" ha due significati: ...
Significato traslato	Virgolette alte	Gli "esperti" non ci prendono mai.
Titoli di libri	Corsivo (`\textit`)	
Titoli di articoli	Virgolette basse	
Nomi di riviste	Corsivo (`\textit`)	

Tabella 5: Scelte stilistiche del presente documento, con alcuni esempi relativi.

siano molto utili e comuni.

B Pacchetti e comandi citati

Si elencano qui di seguito, in ordine alfabetico, i principali pacchetti e comandi LaTeX citati nel testo e utili per comporre un normale testo in italiano seguendo le norme tipografiche discusse.

Pacchetti

[italian]babel — Per usare LaTeX con lingue diverse dall'inglese; imposta la sillabazione italiana, i nomi italiani dei sezionamenti e rende disponibili alcuni comandi utili. § 5.1, § 3.3, § 5.8 e § 2.3.

ellipsis — Corregge la resa tipografica del comando di omissione `\textellipsis`. § 3 e § 5.5.

endnotes — Per avere l'elenco delle note alla fine del documento o alla fine di ogni capitolo invece che a piè di pagina. La documentazione è all'interno del file di stile `endnotes.sty`. § 5.4.

[T1]fontenc — Consigliabile per usare nel documento l'*encoding* standard di LaTeX.

footmisc — Mette a disposizione vari comandi e opzioni per personalizzare le note al testo. L'opzione perpage, per esempio, ricomincia la numerazione delle note a ogni pagina. § 5.4.

indentfirst — Rientra anche il primo paragrafo di un capitolo, secondo l'uso continentale. § 5.3.

[<codifica>]inputenc — Permette di inserire i simboli presenti direttamente da tastiera (richiede in opzione la codifica adatta al sistema operativo usato).

natbib — Pacchetto bibliografico generale, dedicato in particolare all'uso del sistema autore-anno. Si consideri anche l'uso del programma MakeBST. § 5.10.

SIunits — Pacchetto generale per la scrittura corretta delle unità di misura. § 5.8.

ulem — Permette di personalizzare lo stile della sottolineatura. § 1.

xfrac — Migliore resa tipografica per le frazioni, soprattutto se inserite nel testo. Da usare preferibilmente al posto di nicefrac o altri pacchetti di cui rappresenta un superamento. § 5.8.

Comandi

`\,` — Spazio sottile matematico, equivalente a `\thinspace`. § 5.8.

`\ap` — Comando messo a disposizione dall'opzione italian del pacchetto babel per scrivere un'espressione in apice; equivalente a `\textsuperscript`. § 5.8.

`\circ` — Comando da usare in ambiente matematico per inserire un circoletto simile a quello in esponente ai numeri ordinali. § 5.8.

`\footnoterule` — Linea che separa il testo principale dalle note a piè di pagina, inserita subito dopo a `\skip\footins`. § 5.4.

`\skip\footins` — Comando TeX che controlla la distanza verticale (lunghezza elastica) fra la fine del testo e l'inizio delle note a piè di pagina. § 5.4.

`\ignorespaces` — Comando TeX per eliminare spazi indesiderati all'inizio di un testo. § 5.5.

`\ped` — Comando messo a disposizione dall'opzione italian del pacchetto babel per scrivere un'espressione in pedice. § 5.8.

`\sfrac` — Comando messo a disposizione dal pacchetto xfrac per scrivere le frazioni nel testo in modo elegante. § 5.8.

`\textellipsis` — I tre puntini di omissione. § 3 e § 5.5.

`\textsuperscript` — Prende come argomento un simbolo da mettere in posizione di apice. § 5.8.

`\thinspace` — Lo spazio sottile. § 3, § 5.8 e § 5.9.

`\unit` — Comando messo a disposizione dall'opzione italian del pacchetto babel per scrivere correttamente il simbolo delle unità di misura. § 5.8.

Riferimenti bibliografici

Beccari, C. (1991). *La tesi di laurea scientifica*. Hoepli, Milano.

Braams, J. (2005). «Babel, a multilingual package for use with LaTeX's standard document classes». *CTAN*.

Fioravanti, G. (1987). *Il manuale del grafico*. Zanichelli, Bologna.

Flynn, P. (2004). «Typographers' inn». *TUGboat*, **25**, pp. 134–136.

Goossens, M., Mittelbach, F. e Samarin, A. (1994). *The LaTeX Companion*. Addison-Wesley.

Gregorio, E. (2005). «LaTeX: breve guida ai pacchetti più comuni». Gruppo Utilizzatori Italiani di TeX.

Knuth, D. E. (1986). *The TeXbook*. Addison-Wesley.

Lamport, L. (1994). *LaTeX: a Document Preparation System*. Addison-Wesley, 2ª edizione.

Lanza, A. (1992). *Norme grafiche*. De Rubeis, Anzio.

Lesina, R. (1987). *Il manuale di stile*. Zanichelli, Bologna.

Marzullo, M. (2002). «Uso della punteggiatura». Redazione Consulenza Linguistica, *Accademia della Crusca*, `http://www.accademiadellacrusca.it/clic.shtml`.

Meynet, R. (2000). *Norme tipografiche per la composizione dei testi con il computer*. Pontificia Università Gregoriana, Roma. Quinta edizione, prima edizione: Aprile 1995.

Mori, L. (2005). «Scrivere la tesi di laurea con LaTeX 2ε». In *Atti del* GuIt *Meeting 2005*, a cura di Gruppo Utilizzatori Italiani di TeX, Pisa, pp. 69–92.

Salmeri, G. (2001). *Piccolo manuale di stile*. Mneme, `http://mondodomani.org/mneme/gms.htm`.

Serianni, L. (1988). *Grammatica italiana. Italiano comune e lingua letteraria. Suoni, forme, costrutti*. UTET, Torino.

Setti, R. (2002). «Uso delle sigle». Redazione Consulenza Linguistica, *Accademia della Crusca*, `http://www.accademiadellacrusca.it/clic.shtml`.

Trettin, M. (2004). «Elenco dei "peccati" degli utenti di LaTeX 2ε». *CTAN*. Traduzione ed edizione italiana a cura di Emanuele Zannarini.

▷ G. Cevolani
via Marconi 33
41057 Spilamberto
Modena – Italia
`g.cevolani@email.it`

Collegamenti utili del mondo TEX e LATEX

cn: China PR

short name: CTUG
full name: Chinese TeX Users Group
language: chinese
email: info_at_mail.rons.net.cn
web site: www.rons.net.cn
Publication:
title: CTUG Communications, Free Software Magazine
editor: Hong Feng
editor email: fred_at_mail.rons.net.cn
Policy matters:
name: Hong Feng
address: RON's Datacom Co., Ltd.
Suite 3-3, WuZhong Street 200,
DongXiHu District,
Wuhan, Hubei Province,
430040 China P.R.
email: info_at_mail.rons.net.cn
phone: +862783222108
fax: +862783222108

cz: Czech Republic

short name: CSTUG
full name: Československé sdružení živateløu TeXu
language: Czech
members: 380
email: cstug_at_cstug.cz
web site: www.cstug.cz
discussion list: cstex_at_cs.felk.cvut.cz
address: CSTUG c/o FEL ČVUT, Technická 2, 166 27 Praha, Czech Republic
Publication:
title: Zpravodaj CSTUGu
editor: Zdeník Wagner
editor email: bulletin_at_cstug.cz
Banking:
name: Komerèní banka
account: 34735021/0100
address: Václavské námìstí 42, 114 07 Praha 1, Czech Republic
Policy matters:
name: Jaromír Kuben
address: Univerzita obrany, katedra matematiky, Kounicova 65, 612 00 Brno, Czech Republic
email: president_at_cstug.cz
phone: +420 973 443 577
fax: fax.: +420 549 491 820 (Petr Sojka)
General matters:
name: Helena Holovská
address: Matematický ústav AV ČR, Žitná 25, 115 67 Praha 1, Czech Republic
email: secretary_at_cstug.cz
phone: +420 222 090 763
Finance matters:
name: Helena Holovská
address: Matematický ústav AV ČR, Žitná 25, 115 67 Praha 1, Czech Republic
email: treasurer_at_cstug.cz
phone: +420 222 090 763

de: Germany

short name: DANTE e.V.
full name: Deutschsprachige Anwendervereinigung TeX e.V.
language: german
members: 2034
email: dante_at_dante.de
web site: www.dante.de
discussion list: tex-d-l_at_listserv.gmd.de
Publication:
title: Die TeXnische Komödie
editor: Gerd Neugebauer
editor email: dtk-redaktion_at_dante.de
Policy matters:
name: Volker Schaa
address: Postfach 101840
D-69008 Heidelberg
Germany
email: dante-v_at_dante.de
phone: +49622129766
fax: +496221167906
General matters:
name: Günter Partosch
email: secretary_at_dante.de
Finance matters:
name: Tobias Sterzl
email: treasurer_at_dante.de

dk: Denmark

short name: DK-TUG
full name: Danish TeX Users Group
language: danish
members: 84
email: board_at_tug.dk
web site: www.tug.dk
discussion list: dk-tug_at_tug.dk
subscribe at: dk-tug-subscribe_at_tug.dk
address: Department of Mathematical Sciences
Ny Munkegade
DK-8000 Århus C
Denmark
Policy matters:
name: Kaja P. Christiansen
address: Department of Computer Science
University of Aarhus
Aabogade 34
DK-8200 Århus C
Denmark
email: president_at_tug.dk
phone: +4589425713
fax: +4589425624
Finance matters:
name: Arne Jørgensen
address: Gammel Kongevej 7, 1. th.
DK-1610 Copenhagen V
Denmark
email: treasurer_at_tug.dk
phone: +4521650113

ee: Estonia

full name: Estonian User Group
Policy matters:
address: Astrophysical Observatory, Toravere
Enn Saar, Tartu
EE 2444 Estonia
email: saar_at_aai.ee

es: Spain (CervanTeX)

short name: CervanTeX
full name: Grupo de Usuarios de TeX Hispanohablantes
language: Spanish
members: 45
email: secretario_at_cervantex.org
web site: www.cervantex.org
discussion list: es-tex_at_listserv.rediris.es
subscribe at: www.cervantex.org/listas.php
Publication:
title: TeXemplares
editor: Enrique Meléndez
editor email: texemplares_at_cervantex.org
Policy matters:
name: Juan L. Varona
address: University of La Rioja, 26004 Logroño, Spain
email: presidente_at_cervantex.org
General matters:
name: Roberto Herrero Santos
email: secretario_at_cervantex.org
Finance matters:
name: Enrique Meléndez
email: tesorero_at_cervantex.org

esc: Spain (Catalan)

short name: Tirant lo TeX
full name: Catalan TeX Users Group
language: Catalan
web site: www.lsi.upc.edu/~valiente/tug-catalan.html
discussion list: catala-tex_at_ldist.upc.edu
Policy matters:
name: Gabriel Valiente
address: Technical University of Catalonia
Jordi Girona, 1-3
E-08034 Barcelona
Spain
email: valiente_at_lsi.upc.edu

fr: France

short name: GUTenberg
full name: Group francophone des Utilisateurs de TeX
language: french
members: 500
email: gut_at_ens.fr
web site: www.gutenberg.eu.org
Publication:
title: Cahiers GUTenberg
editor: Jacques André
editor email: gut_at_irisa.fr
Policy matters:
name: Maurice Laugier
address: c/o Irisa
Campus universitaire de Beaulieu
F-35042 Rennes cedex
France
email: gut_at_irisa.fr
General matters:
name: Sarah Grimaud
address: Secrétariat GUTenberg
2, rue des Boutons-d'or
F-05000 GAP
France
email: secretariat_at_gutenberg.eu.org
phone: +33681665102
fax: +33492579667
Finance matters:
name: Michèle Jouhet
email: Michele.Jouhet_at_cern.ch

fra: France (Astex)

short name: AsTEX
full name: Association pour la diffusion de logiciels scientifiques liés à TeX
language: french
email: astex-admin_at_univ-orleans.fr
web site: www.univ-orleans.fr/EXT/ASTEX/astex/doc/en/web/html/astex000.htm
discussion list: astex_at_univ-orleans.fr
Policy matters:
name: Michel Lavaud
address: Association AsTEX
BP 6532
45066 Orleans cedex 2
France
email: Michel.Lavaud_at_univ-orleans.fr
phone: +33238640994
Finance matters:
name: Delombera Negga
email: Delombera.Negga_at_wanadoo.fr

gr: Greece

full name: The Greek TeX Friends Group
language: Greek
members: 45
email: eft_at_ocean1.ee.duth.gr
web site: obelix.ee.duth.gr/eft/
Publication:
title: Eutupon
editor: Prof. Basil K. Papadopoulos
Policy matters:
name: Apostolos Syropoulos
address: 366, 28th October Str.
GR-671 00 Xanthi
Greece
email: apostolo_at_obelix.ee.duth.gr
phone: +3054128704

hu: Hungary

short name: MaTeX
full name: Magyar TeX Egyesület
language: Hungarian
members: 15
email: matex_at_math.klte.hu
web site: www.math.klte.hu/~matex/
discussion list: tex-l_at_tigris.klte.hu
subscribe at: www.math.klte.hu/~matex/LevLista.html
address: Inst of Mathematics and Informatics
University of Debrecen
H-4010 Debrecen
P.O. Box 12
Hungary
Policy matters:
name: Antal Járai
address: Department of Numerical Analysis
Eötvös Loránd University
Pázmány Péter sétány 1/D
H-1117 Budapest
Hungary
email: ajarai_at_moon.inf.elte.hu
General matters:
name: Gyöngyi Bujdosó
address: Institute of Mathematics and Informatics
University of Debrecen
H-4010 Debrecen
P.O. Box 12
Hungary
email: ludens_at_math.klte.hu
phone: +36304280403
fax: +3652416857
Finance matters:
name: Deszö Miklós
address: Alfréd Rényi Institute of Mathematics
Hungarian Academy of Sciences,
Reáltanoda 13–15
H-1053 Budapest
Hungary

ie: Ireland

full name: ITALIC
language: irish
discussion list: ITALIC-L_at_listserv.heanet.ie
Policy matters:
name: Peter Flynn
address: University College
Cork
Ireland
email: peter_at_silmaril.ie

in: India

short name: TUGIndia
full name: Indian TeX Users Group
members: 90
email: tugindia_at_tug.org.in
web site: www.tug.org.in
discussion list: tugindia_at_tug.org
subscribe at: www.tug.org/mailman/listinfo/tugindia
address: Floor III, SJP Buildings
Cotton Hills
Trivandrum 695014
India
Publication:
title: TUGIndia
editor email: tij_at_tug.org.in
Banking:
name: HDFC Bank Limited
address: Keston Towers
Vazhuthacaud
Trivandrum 695014
India
Policy matters:
name: K.S.S. Nambooripad
address: Floor III, SJP Buildings
Cotton Hills
Trivandrum 695014
India
email: kssn_at_tug.org.in
phone: +914712337501
fax: +914712333186
General matters:

name: Satish Babu
address: Floor III, SJP Buildings
Cotton Hills
Trivandrum 695014
India
email: sb_at_tug.org.in
phone: +914712337501
fax: +914712333186
Finance matters:
name: K. Anil Kumar
address: Floor III, SJP Buildings
Cotton Hills
Trivandrum 695014
India
phone: +914712337501
fax: +914712333186

is: Iceland

short name: ÍsTeX
full name: Vefur íslenskra TeX notenda
language: Icelandic
web site: rhi.hi.is/istex/
Publication:
title: SamanTeXt
editor: Árni Magnússon
editor email: arnima_at_u.washington.edu
Policy matters:
name: Árni Magnússon
address: Sudurgötu 33
101 Reykjavík
Iceland
email: arnima_at_u.washington.edu

it: Italy

short name: GuIT
full name: Gruppo Utilizzatori Italiani di TeX
language: italian
members: 55
email: guit_at_sssup.it
web site: www.guit.sssup.it
address: c/o Ufficio Statistica
Sant'Anna School of Advanced Studies
Piazza Martiri della Libertà, 33
56127 Pisa, Italy
Publication:
title: ArsTEXnica
editor: Massimiliano Dominici
editor email: arstexnica_at_sssup.it
Policy matters:
name: Maurizio Himmelmann
email: himmel_at_sssup.it
phone: +39 050 883382
General matters:
name: Onofrio 'Ninni' de Bari
email: onodebari_at_gmail.com

kr: Korea

short name: KTUG
full name: Korean TeX User Group
language: korean
email: info_at_mail.ktug.or.kr
web site: www.ktug.or.kr
Policy matters:
name: Kim Kangsu
email: director_at_ktug.or.kr

lt: Lithuania

full name: Lietuvos TeX'o Vartotojų Grupė
members: 8
Policy matters:
name: Vytas Statulevicius
address: Akademijos 4
LT-2600 Vilnius
Lithuania
email: vytass_at_ktl.mii.lt
phone: +3702359609
fax: +3702359804

mx: Mexico

full name: TeX México
language: Spanish
email: tex_at_linuxopensource.com.mx
web site: linuxopensource.com.mx/tex/
Policy matters:
name: Cuauhtémoc Pacheco Díaz
address: Rayon No. 523, Centro 58000
Morelia, Michoacan
Mexico
email: temo_at_ibiblio.org
phone: +52143128724
fax: +52143173945

nl: Netherlands, Belgium

short name: NTG
full name: Nederlandstalige TeX Gebruikersgroep
language: dutch
members: 300
email: info_at_ntg.nl
web site: www.ntg.nl
discussion list: tex-nl_at_ntg.nl
subscribe at: LISTSERV_at_nic.surfnet.nl
address: Maasstraat 2
5836 BB Sambeek
The Netherlands
Publication:
title: MAPS
editor: Wybo Dekker
editor email: maps_at_ntg.nl
Banking:
name: Postbank
account: 1306238
(swift) code: PSTBNL21
routing no.: NL05PSTB0001306238
address: Amsterdam
Policy matters:
name: Hans Hagen
address: Pragma
Ridderstraat 27
8061 GH Hasselt
The Netherlands
email: ntg-president_at_ntg.nl
phone: +31384775369
fax: +31384775374
General matters:
name: Willi Egger
address: Maasstraat 2
5836 BB Sambeek
The Netherlands
email: ntg-secretary_at_ntg.nl
phone: +31485573896
Finance matters:
name: Wybo Dekker
address: Deilsedijk 60
4158 CH Deil
The Netherlands
email: ntg-treasurer_at_ntg.nl
phone: +31345652164

no: Nordic countries

full name: Nordic TeX Users Group
language: Nordic languages
members: 78
web site: www.ifi.uio.no/~dag/ntug/
discussion list: nordictex_at_ifi.uio.no
Policy matters:
name: Dag Langmyhr
address: University of Oslo
PO Box 1080 Blindern
N-0316 Oslo
Norway
email: dag_at_ifi.uio.no
phone: +4722852450
fax: +4722852401

ph: Philippines

short name: TUG-Philippines
full name: Philippines TeX Users Group
Policy matters:
name: Felix P. Muga II
address: Ateneo de Manila University
Loyola Heights
Quezon City
Philippines
email: fpmuga_at_admu.edu.ph
phone: +6324266001 ext 2515
fax: +6324266008

pl: Poland

full name: Polska Grupa Użytkowników Systemu TeX – GUST
language: Polish
members: 276
email: sekretariat_at_gust.org.pl
web site: www.GUST.org.pl
discussion list: gust-l_at_man.torun.pl
address: Polska Grupa Użytkowników Systemu TeX
Pl. Rapackiego 1
PL-87-100 Toruń
Poland
Publication:
title: Biuletyn GUST
editor: Tomasz Przechlewski
editor email: ekotp_at_univ.gda.pl
Banking:
name: BIG Bank Gdański SA, OddziałMillenium, Toruń
account: 80 11602202 0000 0000 5530 4981
Policy matters:
name: Jerzy Ludwichowski
address: Information & Communication Technology Centre
Nicolaus Copernicus University
Pl. Rapackiego 1
PL-87-100 Toruń
Poland
email: prezes_at_gust.org.pl
phone: +48 56 6112742
fax: +48 56 6221850
Finance matters:
name: Jolanta Szelatyńska
address: Information & Communication Technology Centre
Nicolaus Copernicus University
Pl. Rapackiego 1
PL-87-100 Toruń
Poland
email: secretary_at_gust.org.pl
phone: +48 56 6112741
fax: +48 56 6221850

pt: Portugal

short name: GUTpt
full name: Grupo de Utilizadores de TeX
language: Portuguese
web site: gentzen.mat.uc.pt/~gutpt/
discussion list: gutpt_at_gentzen.mat.uc.pt
Policy matters:
name: Pedro Quaresma de Almeida
address: Coimbra University, Dep. Matemática
Largo D.Dinis
Apartado 3008
3001-454 COIMBRA
Portugal
email: pedro_at_mat.uc.pt
phone: +351239791181

ru: Russia

short name: CyrTUG
full name: Associaciia Polźovatelei Kirillicheskogo TeXá
email: cyrtug_at_mir.msk.su
web site: www.cemi.rssi.ru/cyrtug/
Policy matters:
name: Eugenii V. Pankratiev
email: cyrtug_at_cemi.rssi.ru
General matters:
name: Irina Makhovaya, Executive Director
address: Mir Publishers
2, Pervyi Rizhskii Pereulok
Moscow 129820
Russia
email: irina_at_mir.msk.su
phone: +70952860622, +70952861777
fax: +70952889522

si: Slovenia

full name: TeXCeH
web site: vlado.fmf.uni-lj.si/texceh/texceh.htm
Policy matters:
name: Vladimir Batagelj
address: Jadranska 19
SI-61111 Ljubljana
Slovenia
email: Tex.Ceh_at_fmf.uni-lj.si

uk: United Kingdom

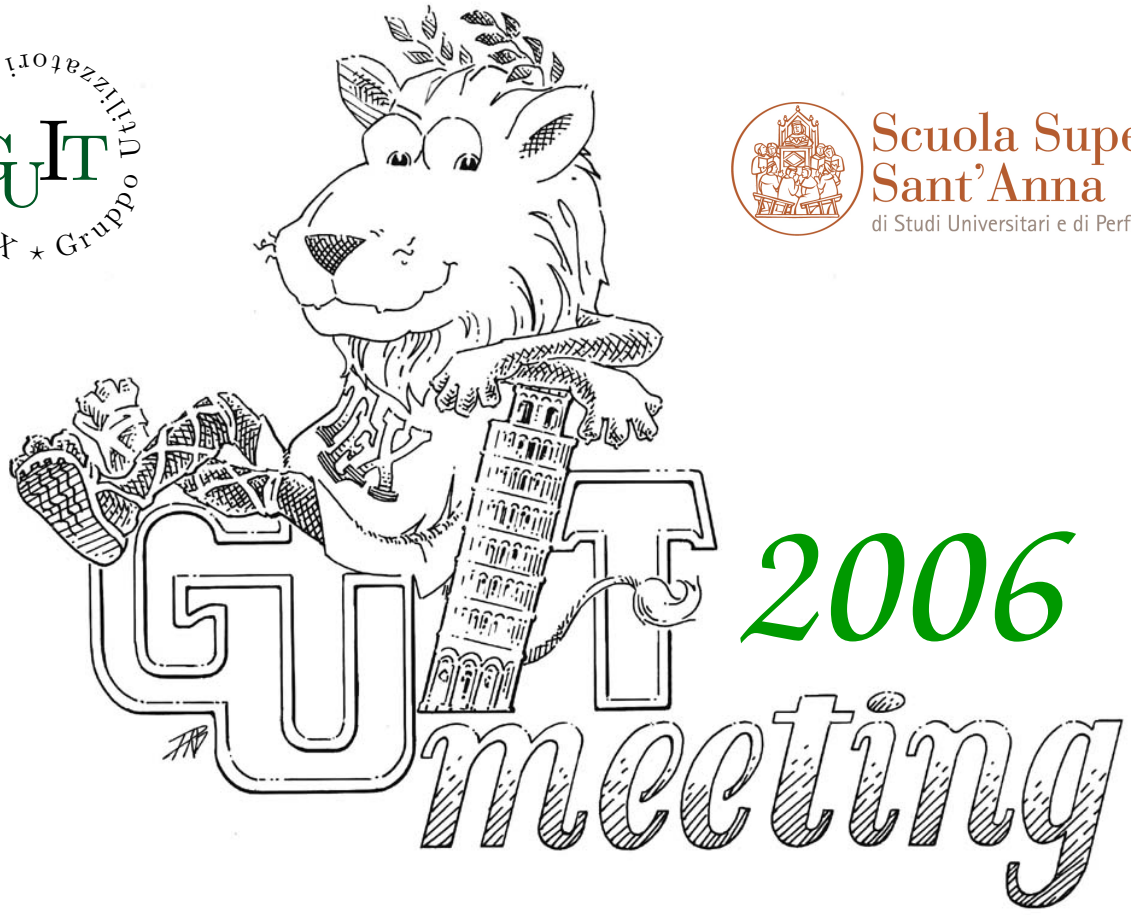
short name: UKTUG
full name: UK TeX Users' Group
language: British English
members: 150
email: uktug-enquiries_at_tex.ac.uk
web site: uk.tug.org
subscribe at: uk.tug.org/Membership/
address: UKTUG (membership)
61 Victor Road
Kensal Green
London
NW10 5XB
UK
Publication:
title: Baskerville (dormant)
editor: tba
editor email: baskerville_at_tex.ac.uk
Policy matters:
name: Jay Hammond
email: uktug-chairman_at_tex.ac.uk
General matters:
name: David Osborne
address: UKTUG secretary c/o
61 Victor Road
Kensal Green
London
NW10 5XB
UK
email: uktug-secretary_at_tex.ac.uk
Finance matters:
name: Tobias Wahl
email: uktug-enquiries_at_tex.ac.uk

us: TeX User Group (international)

short name: TUG
full name: TeX Users Group
members: 1800
email: office_at_tug.org
web site: www.tug.org
discussion list: board_at_tug.org
address: TeX Users Group
P.O.Box 2311
Portland, OR 97208-2311
USA
Publication:
title: TUGboat
editor: Barbara Beeton
editor email: TUGboat_at_tug.org
Policy matters:
name: Karl Berry
address: 1466 Naito Ave NW, Suite 3141
Portland, OR 97209
USA
email: president_at_tug.org
phone: +15032239994
fax: +12062033960
General matters:
name: Sue DeMeritt
email: secretary_at_tug.org
Finance matters:
name: David Walden
email: treasurer_at_tug.org

vn: Vietnam

short name: VietTUG
full name: Vietnamese TeX Users Group
language: vietnamese
email: kyanh_at_o2.pl
web site: www.viettug.org



Terzo convegno nazionale su TEX, LATEX e tipografia digitale

Pisa, 21 ottobre 2006

Aula Magna, Scuola Superiore Sant'Anna

Call for Paper

Sabato 21 ottobre 2006 presso l'Aula Magna della Scuola Superiore Sant'Anna di Pisa si terrà il terzo Convegno annuale su TEX, LATEX e tipografia digitale organizzato dal Gruppo Utilizzatori Italiani di TEX. Il Convegno sarà un momento di ritrovo e di confronto per la comunità LATEX italiana, tramite una serie di interventi atti sia a contribuire all'arricchimento sia a supportarne lo sviluppo
Maggiori informazioni sul Convegno e sulle modalità di presentazione degli interventi sono disponibili all'indirizzo:

`http://www.guit.sssup.it/guitmeeting/2006/`

ArsTEXnica

Rivista italiana di TEX e LATEX

Numero 1, Aprile 2006

3 Editoriale
Massimiliano Dominici

5 Intervista a Donald Knuth
Gianluca Pignalberi

8 I registri *token*: questi sconosciuti
Claudio Beccari

14 Ridefinire i comandi primitivi TEX e applicazioni a LATEX
Enrico Gregorio

20 L'ambiente picture
Massimo Caschili

29 Norme tipografiche
Gustavo Cevolani