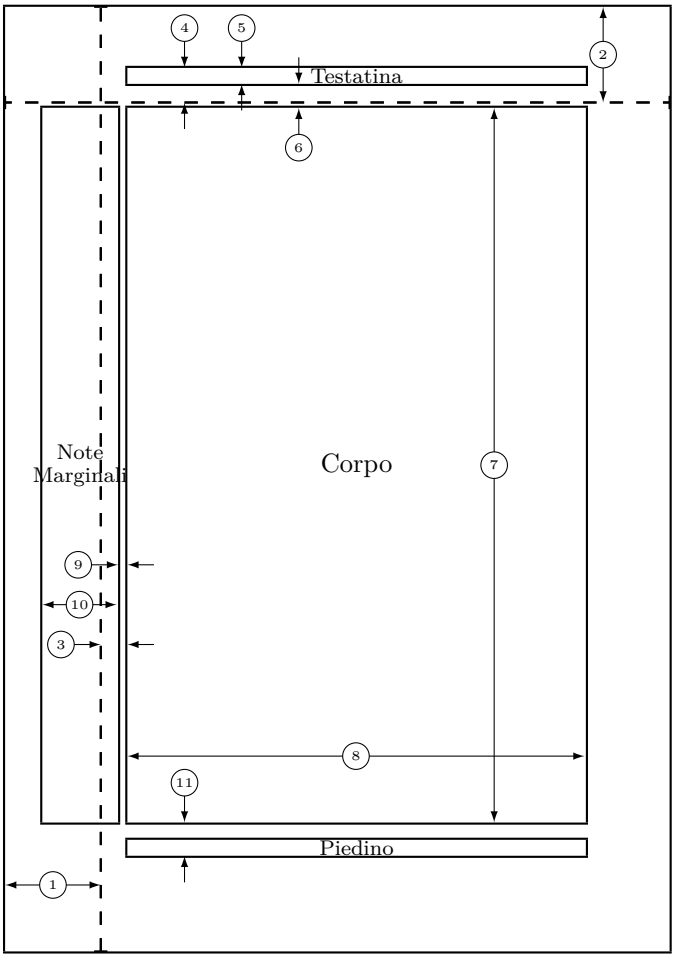
Gruppo Utilizzatori Italiani di TEX

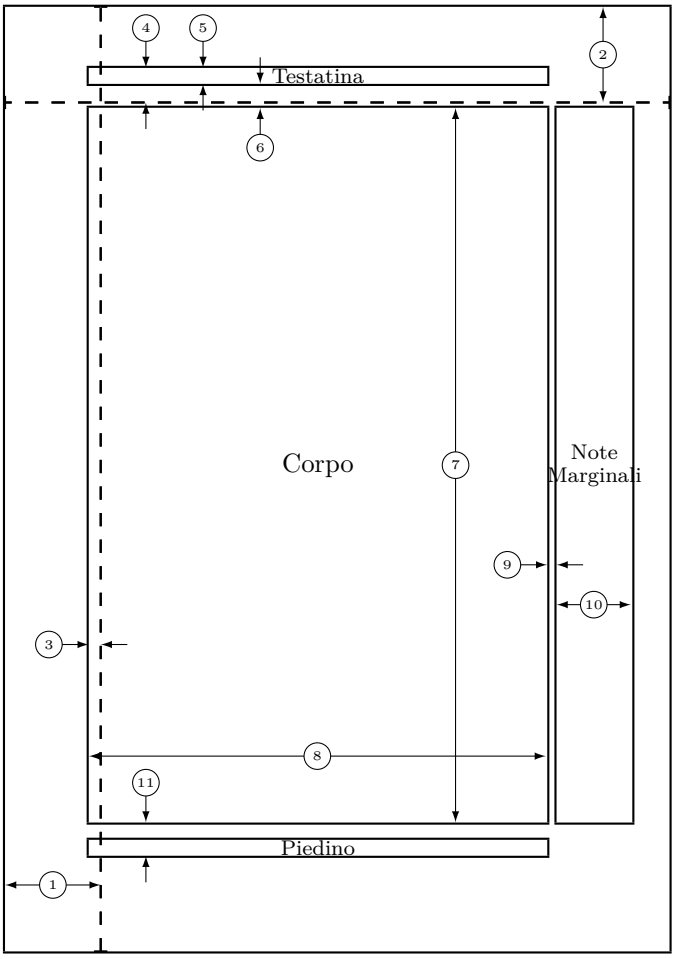
Introduzione all'arte della composizione tipografica con LATEX

Versione B5-0.99.56 — 2024-09-19
Pdf minor version = 5

Introduzione all'arte della composizione tipografica con LaTeX



1	un pollice + \hoffset	2	un pollice + \voffset
3	\evensidemargin = 20pt	4	\topmargin = -26pt
5	\headheight = 12pt	6	\headsep = 18pt
7	\textheight = 538pt	8	\textwidth = 345pt
9	\marginparsep = 7pt	10	\marginparwidth = 57pt
11	\footskip = 25pt		\marginparpush = 5pt (non mostrato)
	\hoffset = 0pt		\voffset = 0pt
	\paperwidth = 500pt		\paperheight = 711pt



1	un pollice + \hoffset	2	un pollice + \voffset
3	\oddsidemargin = -9pt	4	\topmargin = -26pt
5	\headheight = 12pt	6	\headsep = 18pt
7	\textheight = 538pt	8	\textwidth = 345pt
9	\marginparsep = 7pt	10	\marginparwidth = 57pt
11	\footskip = 25pt		\marginparpush = 5pt (non mostrato)
	\hoffset = 0pt		\voffset = 0pt
	\paperwidth = 500pt		\paperheight = 711pt

Introduzione all'arte della composizione tipografica con LaTeX

GuIt

Versione B5-0.99.56 — 2024-09-19
Pdf minor version = 5

La foto 2.2 della pagina 50 e le foto delle pagine 53, 277, e 507 — 509 sono di Claudio Beccari.
L'immagine 2.1 della pagina 50 è stata ricavata da un indirizzo Internet non più reperibile; l'autore sembra essere Hermann Zapf.

Coordinatore: Claudio Beccari

Hanno collaborato a questo testo in modo diretto o indiretto: Alessandro Andretta, Luciano Battaia, Francesco Biccari, Paolo Biffis, Riccardo Campana, David Carlisle, Antonio Cervone, Gustavo Cevolani, Agostino De Marco, Roberto Giacomelli, Tommaso Gordini, Enrico Gregorio, Orlando Iovino, Maurizio Himmelmann, Jerónimo Leal, Antonio Macrì, Giuseppe Molteni, Federico Morchio, Lorenzo Pantieri, Gianluca Pignalberi, Ottavio Rizzo, Salvatore Schirone, Luigi Scarso, Andrea Tonelli, Ivan Valbusa, Filippo Vomiero, Emanuele Zannarini.

Un grazie particolare a Enrico Gregorio.

Presentazione

[1.1]Questo testo è rivolto a coloro che amano la bella composizione tipografica e trovano che i testi composti con LaTeX siano molto professionali. Questo implica che il lettore tipo abbia già un certa dimestichezza con LaTeX e abbia letto, almeno in parte, o il testo introduttivo [53] o, meglio ancora, l'ottimo manuale di Lorenzo Pantieri e Tommaso Gordini *L'arte di scrivere con LaTeX*, [55].

Alcune parti di questo testo possono risultare un po' ostiche a chi è completamente digiuno di programmazione, ma si è fatto ogni sforzo per rendere comprensibile queste parti anche a chi ha poca dimestichezza con il calcolatore. Certo c'è un compromesso fra chiarezza e concisione, ma si spera che il lettore vorrà ugualmente comprendere l'impegno del gruppo di persone che ha dato vita a questo testo.

Questo testo sulla composizione tipografica mediante LaTeX è stato infatti predisposto da diversi membri del Gruppo degli Utilizzatori Italiani di TeX, GuIt, il cui obiettivo è proprio quello di far conoscere il sistema di composizione sviluppato dal matematico Donald E. Knuth alla fine degli anni '70. Non si tratta di un programma di interesse archeologico, perché esso è vivo e vegeto, ha dato origine a un buon numero di discendenti e alcune sue parti sono usate all'interno di altri programmi di elaborazione di testi, senza che gli utenti di questi programmi lo sappiano.

Il sistema TeX è stato uno dei primi esempi di software libero; questa sua qualità ne ha permesso il contributo creativo e/o critico di una moltitudine di utenti, come succede sempre con il software libero, per cui si è arricchito nel tempo di una moltitudine di estensioni che gli permettono di comporre praticamente qualsiasi cosa, tranne, forse, certi tipi di pieghevoli pubblicitari.

Si tratta di un programma di composizione tipografica, non di un impaginatore; quest'ultimo tipo di programmi consente di agire sul materiale da impaginare come se fosse una figura da modificare o da adattare, anche se svolge in parte le funzioni di compositore.

LaTeX, una specie di linguaggio di programmazione del sistema TeX, è, appunto, un linguaggio che va interpretato da appositi programmi del sistema TeX; la sinergia del linguaggio e dei programmi interpreti consente di comporre tipograficamente dei testi contenenti testo corrente, sia in prosa sia in poesia, scritti in qualunque alfabeto, per esempio latino, greco, cirillico, dall'andamento diretto (da sinistra a destra), oppure ebraico, arabo, dall'andamento inverso

(da destra a sinistra), oppure cinese, giapponese, coreano, dall'andamento anche verticale.

Questa sinergia può gestire font di ogni genere, sia quelli a matrici di punti, sia i font PostScript, sia i font TrueType, OpenType, eccetera. Tali font possono essere codificati in varie maniere, ma alcuni 'figli' di TEX gestiscono anche i font codificati secondo la norma UNICODE.

La caratteristica che più differenzia il sistema TEX dagli altri elaboratori di composizione è il fatto che per comporre un documento con questo sistema bisogna agire in tempi diversi per introdurre il testo e per comporlo; in questo non è molto diverso da certi procedimenti professionali di impaginazione, dove il testo da comporre viene introdotto in un file di solo testo che poi viene in un secondo tempo fatto fluire dentro il programma di impaginazione, assemblandolo insieme alle figure e all'altro materiale non testuale per generare il documento 'finito', pronto da inviare alla fotoincisione e alla stampa.

Invece questo modo di comporre è molto diverso da quello dei word processor, dove il compositore vede direttamente sullo schermo del suo elaboratore il testo già composto, così da poter esaminare immediatamente il frutto del suo lavoro; l'analisi di questa differenza nel modo di procedere verrà svolta nel primo capitolo.

Il secondo capitolo darà al lettore alcune nozioni di tipografia, se non altro per abituarlo ad alcune parole che ricorrono spesso nella descrizione delle varie operazioni compositive. Chi avesse già queste nozioni può saltare la lettura di questo capitolo, ma se, nonostante tutto, decidesse di leggerlo, potrebbe constatare che alcune nozioni e/o alcuni vocaboli in questa guida sono usati per indicare cose leggermente diverse da quelle che conosceva.

Il terzo capitolo è dedicato all'ortografia tipografica; oltre alla definizione, che ovviamente non contrasta con l'ortografia della lingua, vengono impostate anche le buone regole per l'uso della cesura in fin di riga, delle forme dei caratteri, dell'uso della punteggiatura, dell'elisione e del troncamento, sottolineando che la grammatica consente numerose varianti, ma la buona tipografia predilige l'omogeneità e l'adesione a consuetudini ormai consolidate; la buona tipografia aiuta il lettore a leggere senza affaticarsi; usare convenzioni diverse dalle 'solite' affatica il lettore che deve abituarsi a comprendere uno scritto dall'aspetto per lui insolito.

Il quarto capitolo esaminerà le procedure da seguire per procurarsi il software del sistema TEX e per installare i programmi e i file accessori; non si vuole mettere in ombra la centralità dei programmi di composizione del sistema TEX, ma bisogna attrarre l'attenzione sui programmi accessori, per esempio gli *shell editor* la cui scelta può risultare determinante nella facilità con cui si può ottenere il meglio dal sistema TEX.

I capitoli cinque, sei, sette ed otto richiamano i primi rudimenti della programmazione LATEX, che in modo asincrono attraverso i programmi interpreti diventa il testo composto. Si parlerà essenzialmente di composizione testuale, di liste e di tabelle. Queste ultime già possono dare qualche pensiero anche al lettore con una modesta esperienza alle spalle.

La programmazione della composizione mediante LaTeX si distingue da altri procedimenti di composizione per consentire al compositore, oltre che allo scrittore, di concentrarsi sul messaggio da trasmettere al lettore, invece che sulla forma da dargli, cioè sulla sua estetica. A seconda del documento da comporre lo stile di composizione può essere molto diverso; anche la semplice impaginazione può assumere aspetti grafici diversi; LaTeX consente di scegliere in modo globale il tipo di documento e di apportare piccole modifiche stilistiche durante la composizione, senza che il compositore debba preoccuparsi della nerezza dei caratteri, oppure della distanza del numero indicativo di un paragrafo dall'inizio del suo titolino. A tutte queste cose pensa la scelta iniziale del tipo di documento e specificando opportune opzioni. Il capitolo sei si occupa appunto di queste cose.

I capitoli nove e dieci si dedicano alla creazione, manipolazione e inclusione di figure.

Passando ad argomenti più specializzati, i capitoli undici e dodici si occupano della preparazione della bibliografia e della composizione di indici analitici e glossari.

I capitoli tredici e quattordici si dedicano invece alla composizione della matematica; se il lettore ha familiarità con manuali di tipografia, avrà notato che l'argomento della composizione della matematica è praticamente assente da quei manuali, perché si tratta di un tipo di composizione abbastanza specializzato e che interviene raramente nei libri pubblicati. Ma un documento non è necessariamente un libro di narrativa; può essere un rapporto tecnico, una tesi di laurea o di dottorato, un manuale o un prontuario tecnico, un articolo scientifico, insomma, uno scritto dove la matematica compare, spesso, anche in forma avanzata.

Alla composizione di testi letterari e di filologia è dedicato il capitolo quindici; questi testi verosimilmente non contengono una sola formula matematica, ma richiedono un tipo di composizione spesso bidimensionale, che, come la matematica, richiede di comprendere a fondo i meccanismi compositivi bidimensionali. Alle necessità compositive della filologia si provvede mediante l'uso di moduli esterni, chiamati *file* o *pacchetti di estensione*, che fanno parte integrante del sistema TeX.

Il capitolo sedici si riferisce alla preparazione delle presentazioni, cioè di quei documenti che verranno verosimilmente usati per essere proiettati mentre se ne espone il contenuto a voce durante una conferenza, una lezione, o simili.

Il capitolo diciassette tratta dell'arte della composizione tipografica con LaTeX; questo è proprio il capitolo che giustifica il titolo di questo manuale; vi si parlerà delle tecniche di cui LaTeX o, più ancora, i programmi di composizione del sistema TeX dispongono per eseguire la composizione tipografica come e meglio della totalità dei word processor e meglio di molti programmi di impaginazione; si tratta della microgiustificazione, una tecnica a cui ricorrevano fin dall'inizio i prototipografi, a cominciare da Gutenberg, ma di cui si è un po' persa l'abitudine nel corso dei secoli successivi, sia per i costi, sia per i vincoli imposti dai caratteri metallici; per fortuna la composizione elettronica è svincolata da queste limitazioni, ma ovviamente bisogna che il programma di composizione sia stato predisposto per la microgiustificazione.

Il capitolo diciotto si riferisce ai caratteri da stampa e ne descrive le particolarità, i comandi per gestirli, i modi per caricare altri caratteri; fornisce qualche nozione di tipografia, ma dedica ampio spazio ai font del sistema TEX, il quale, è bene ribadirlo, non è limitato ai font disponibili in prima installazione, che sono già molto numerosi, ma può usare virtualmente qualsiasi font disponibile sia gratuito, sia commerciale.

Il capitolo diciannove è dedicato all'arte di definire nuovi comandi compositivi per agevolare il compito del compositore.

Il capitolo venti tratta invece del progetto grafico delle pagine e delle strutture che compongono il documento; normalmente queste sono decisioni e scelte operate dal grafico editoriale; con LATEX spesso siamo noi stessi i grafici editoriali, i compositori, talvolta anche gli editori dei nostri documenti; è bene avere anche una buona formazione in questo senso, almeno fino al livello di conoscere i problemi, senza avere la presunzione di saperli risolvere come un professionista. Conoscendo i problemi, però, può nascere il desiderio di conoscere più a fondo l'arte tipografica, che ci porterà non solo ad apprezzare maggiormente i prodotti tipografici ben riusciti, ma anche ad arricchire la nostra cultura in un settore spesso trascurato o considerato erroneamente troppo tecnico per meritare il nome di 'arte'.

Ricordiamo invece che il nome del sistema TEX è ottenuto dalle prime tre lettere (maiuscole) della parola greca τέχνη[1], che, come ricorda Knuth stesso, vuol dire *arte*.

Questo testo non è un manuale; per LATEX ce ne sono di liberi e di commerciali assai validi; a questo argomento è dedicato il capitolo ventuno. Nello stesso tempo, dicendo che questo non è un manuale si vorrebbe sottolineare che le indicazioni qui esposte servono per affrontare la composizione con LATEX con un approccio che guarda di più alla sostanza, alla composizione professionale, che non all'aspetto grafico del singolo dettaglio man mano che questo si presenta, come succede spesso, invece, con i vari word processor.

Per esempio in questo testo si parla di 'scatole' proprio di sfuggita, mentre in ogni manuale questo argomento richiede almeno una sezione espressamente dedicata loro. Così si parla poco o nulla di contatori o di lunghezze rigide o elastiche; sono argomenti importanti, ma che servono maggiormente per scrivere i programmi, le macroistruzioni, contenute nei file che specificano lo stile compositivo o nei file che raccolgono le macro personali. Durante la composizione non bisognerebbe mai perdersi in questi dettagli.

Piuttosto la programmazione in linguaggio 'nativo' TEX può diventare essenziale per l'utente che deve comporre testi dallo stile insolito o che abbia bisogno di strutture compositive particolari.

Qui, però, tranne una esposizione sommaria nel capitolo 19, non si parlerà di queste cose, ma si rinvia direttamente il lettore a manuali che trattano questo aspetto con maggiore dettaglio, (capitolo 21). Si veda piuttosto il capitolo 29, di

[1] La parola τέχνη in Italia viene pronunciata 'tècne'; in Grecia 'téchni'; in ogni caso quella che sembra una X nel nome di TEX e di LATEX è una χ maiuscola e va pronunciata come una 'k' o come i toscani pronunciano la 'c' iniziale di 'casa', non come una 'x'.

tipo più manualistico, dove sono raccolte, commentate, modificate le informazioni che Leslie Lamport ha scritto nel suo manuale [39].

Il capitolo ventidue è dedicato ad un aspetto piuttosto moderno e fortemente legato alla tipografia elettronica, vale a dire si occupa del problema dell'archiviabilità dei documenti elettronici; questa caratteristica richiede strumenti ed accorgimenti particolari che si possono raggiungere con l'uso del sistema TEX, ma che richiedono comunque una speciale attenzione.

Il capitolo ventitré riguarda invece un aspetto produttivo che coinvolge la tipografia elettronica; si tratta del lavoro di assemblaggio dei file da comporre con il sistema TEX, quando i file iniziali non hanno il mark-up di LATEX. Si può eseguire l'elaborazione manuale, ma è meglio disporre di adeguati strumenti per la conversione automatica, almeno per ottenere un primo approccio al mark-up di LATEX, ma con la trasformazione delle codifiche dei vari caratteri in modo da non alterare il messaggio del testo da comporre e, specialmente, per non introdurre errori durante il processo di conversione. L'editoria moderna deve poter usare questi strumenti con cognizione di causa, sia per il lavoro commerciale sia per quello che gradatamente si va estendendo, cioè la documentazione libera, la produzione di testi liberamente scaricabili dalla rete, ma composti con ogni cura al massimo livello qualitativo.

Il capitolo 24 è dedicato alla simbologia e alla nomenclatura delle grandezze fisiche, nonché ai simboli codificati dalle norme ISO per l'uso nella matematica usata dai fisici, dai tecnologi e in generale dagli scienziati che misurano grandezze; vi sono anche non pochi riferimenti al Sistema Internazionale delle Unità di Misura e alla loro 'ortografia'.

Il capitolo 25 spiega invece in ogni dettaglio l'algoritmo che viene usato dai programmi del sistema TEX per dividere le parole in fin di riga; questo è un procedimento che in italiano funziona impeccabilmente nella totalità dei casi, ma è bene conoscerne i dettagli per intervenire in quei pochi casi in cui la giustificazione non viene eseguita perfettamente oppure quando si scrive in una lingua straniera dalle regole più complesse di quelle che valgono per l'italiano; oppure per ricorrere ad una sillabazione italiana diversa da quella prescritta dalla norma UNI, ma consentita in casi particolari.

Il capitolo 26 vorrebbe riempire un vuoto di informazione che per usare LATEX va assolutamente colmato. Si tratta della distinzione fra la codifica di entrata e quella di uscita; fra la codifica che l'utente, tramite i suoi programmi, usa per scrivere un testo usando la tastiera, e quella che il programma di composizione usa in relazione ai font impiegati. Bisogna conoscere queste due codifiche, distinguerle e usarle con cognizione di causa.

Il capitolo 27, molto tecnico, cerca di spiegare i dettagli di come ogni motore di composizione del sistema TEX costruisca i capoversi e costruisca le pagine; si tratta di un meccanismo abbastanza complesso, ma la sua comprensione aiuta a capire perché la composizione talvolta sia diversa da come ce la aspetteremmo. Normalmente ha ragione il motore di composizione a fare ciò per cui è stato programmato, ma talvolta può essere necessaria una piccola dose di aiuto consapevole.

Il capitolo 29, lungi dal rappresentare una semplice traduzione della corrispondente appendice del manuale [39] di Leslie Lamport, il creatore di LaTeX, pur mantenendone l'impostazione, rappresenta, si spera, una fonte comoda e pratica per rivedere in modo ragionato ma non troppo prolisso la sintassi dei vari comandi del mark-up di LaTeX. In questa parte, per la verità, si sono ridotte al minimo rispetto al manuale di Lamport, intere parti che oggi sono considerate superate; invece alcune altre parti sono state estese considerevolmente.

Il capitolo 30 tratta dei continui aggiornamenti di LaTeX. Questo testo è andato formandosi nell'arco di diversi anni, con i contributi di molte persone, che hanno dedicato tempo e esperienza per fornire informazioni esatte e aggiornate. Tuttavia dal 2018 gli aggiornamenti del sistema TeX sono continui e molto importanti; mentre il curatore di questo testo cerca di apportare le modifiche al contenuto degli altri capitoli, in questo particolare riporta le informazioni pubblicate dal LaTeX Project Team attraverso le *news letters* che quel gruppo di lavoro del TUG (TeX Users Group) pubblica in seguito all'introduzione di correzioni, modifiche, estensioni che esso apporta in continuazione. Il curatore cerca di selezionare le notizie di queste *news letters* in modo da riportare nel capitolo 30 quelle che possono interessare l'utente finale; il lettore attento, specialmente se desidera usare le novità introdotte nel costruire i suoi file di classe o i suoi pacchetti di macro personali, può leggere ogni *news letter* usando la funzionalità della sua installazione consistente nel dare da terminale il comando `texdoc ltnews`⟨*numero*⟩ in modo da leggere l'intera *news letter* identificata da quel ⟨*numero*⟩.

Il lettore non si scoraggi se in questo testo vengono usate molte parole inglesi; in un certo senso la cosa è voluta, perché i comandi del *mark-up* di LaTeX sono quasi tutti formati da parole inglesi o da loro abbreviazioni; in ogni punto in cui questi comandi vengono descritti si è sempre cercato di associare le parole inglesi alla spiegazione italiana; tuttavia non si è rifuggiti da alcune similitudini insolite, come 'flottante' (che esiste in italiano con un significato simile) corrispondente a *floating*, ritenuto preferibile rispetto all'italianissima parola 'mobile' che esprime lo stesso concetto, ma che forse è espresso con una parola slegata dai messaggi che LaTeX espone sullo schermo durante la sua esecuzione. Così si è frequentemente usata la locuzione *di default*, piuttosto che usare espressioni italiane come 'predefinito', 'da usarsi in mancanza d'altro', 'se non si è specificato diversamente', e simili locuzioni quasi equivalenti l'una all'altra.

Inoltre, sebbene non si sia sicuri di essere stati sempre consistenti, è stata usata la congiunzione oppositiva 'o' per indicare il concetto esclusivo di 'o questo o quello ma non entrambi', e la congiunzione 'oppure' per indicare il concetto inclusivo di 'o questo o quello o entrambi'. La lingua parlata e quella scritta sono flessibili e talvolta non è necessario essere così precisi, ma si cerchi di tenere presente questo particolare uso delle congiunzioni oppositive.

Di fianco a molte figure vi è una piccola legenda, spesso in verticale, con il nome dell'autore e il nome del programma, quasi sempre del sistema TeX, che l'autore ha usato. Questa indicazione non è messa vicino alle fotografie oppure

alle immagini tratte da Internet; la paternità di queste immagini è indicata, ove possibile, nel retrofrontespizio. Per le altre figure, oltre a indicare l'autore del disegno mostrato, si è ritenuto utile indicare anche il programma usato per realizzare il disegno; quasi tutti i disegni sono stati realizzati usando comandi e mark-up forniti da pacchetti di estensione del programma di composizione; questi pacchetti consentono di eseguire egregiamente molti tipi di disegni, senza bisogno di ricorrere a programmi esterni; hanno anche il vantaggio di essere stilisticamente compatibili con la tipografia eseguita con LaTeX, cosa che spesso non è possibile ottenere con i normali word processor o con altri programmi di disegno assistito dal calcolatore.

Il materiale di questa guida è stato riordinato rispetto alle prime edizioni; ovviamente è anche stato aggiornato e l'aggiornamento continuo cerca di non restare indietro rispetto all'evoluzione continua del sistema TeX; non è possibile garantire che questo testo sia sempre aggiornato all'ultima distribuzione del sistema TeX, ma si fa il possibile per farlo. Si veda più avanti anche quanto scritto nell'introduzione del capitolo 30.

Questa guida può essere composta in formato A4 (210 mm per 297 mm) o in formato B5 (176 mm per 250 mm); quest'ultimo formato può venire stampato e rifilato in modo che il volume brossurato a colla o legato possa trovare facilmente spazio nei normali scaffali; può venire anche stampato in tre tomi distinti in formato B5, dove però le pagine iniziali (dal frontespizio fino agli indici) e le pagine finali (dalla bibliografia in poi) siano comuni a tutti i tomi; i riferimenti degli indici generale e analitici saranno quindi sempre corretti anche in riferimento a tomi diversi.

Indice

Elenco delle tabelle

Elenco delle figure

Avvertenze

In questo testo si parla di quello che si può fare con la ricca dotazione di programmi e di pacchetti presenti in ogni distribuzione del sistema TEX. Le distribuzioni del sistema TEX gratuite sono essenzialmente due, TEX Live e MiKTEX. In commercio ci sono diverse altre distribuzioni.

Nel sito `http://www.tug.org"/store/` si dice che l'immagine ISO del disco può essere scaricata gratuitamente. L'immagine ISO (che con i sistemi operativi recenti può essere usata come un vero DVD) può poi essere masterizzata su un DVD vergine e usata tranquillamente per conservale quanto distribuito. La distribuzione scaricata dal sito TUG menzionato è completa e permette l'installazione sulle macchine Windows, Linux; nel sito indicato c'è una sezione dedicata alle macchine Mac con sistema operativo OSX con processore a 64 bit; L'installazione completa su una macchina Mac è molto veloce: scaricare il file `MacTeX.pkg` dura più o meno a lungo a seconda della velocità della rete Internet e del traffico in rete; scaricare il contenuto del pacchetto suddetto per installare il sistema TEX completo dura pochi minuti. Indipendentemente dalle particolari piattaforme, ognuna viene dotata del software necessario per aggiornare la propria installazione con la frequenza che preferisce; chi scrive aggiorna ogni 7 o 10 giorni e consigli di farlo a tutti i lettori di questa guida.

Anche la distribuzione MiKTEX (valida solo per le macchine Windows) può venire scaricata gratuitamente dal sito `http://miktex.org/`. Più precisamente da questo sito è possibile scaricare dei semplici programmi di installazione, uno molto piccolo per installare una versione di base e uno di dimensioni maggiori per scaricare la distribuzione completa. Benché la distribuzione di base possa essere arricchita in modo praticamente automatico via via che l'utente richiama funzionalità ancora non installate, si consiglia vivamente di eseguire l'installazione completa.

Nel seguito si farà riferimento specialmente alla distribuzione TEX Live, perché con minime varianti è identica per tutte le piattaforme di elaborazione più comuni. Anzi, anche per chi lavora su macchine Windows, sebbene esista l'eccellente distribuzione MiKTEX, espressamente prodotta per lavorare con i vari sistemi operativi della Microsoft, si consiglia di installare la versione TEX Live. Certe personalizzazioni necessarie per la composizione tipografica di alta qualità diventano relativamente difficili da fare con la distribuzione MiKTEX, prevista per utenti ai quali, lavorando con un sistema operativo "proprietario", non sono concesse troppe personalizzazioni.

Nel seguito quando si scriverà CTAN ci si riferirà al Comprehensive TEX Archive Network (la rete degli archivi completi del sistema TEX); questa rete è formata da due server principali dislocati in varie parti del mondo e da un numero sterminato di "mirror" che ne rispecchiano il contenuto. I primi sono aggiornati in modo sincrono ogni notte; i secondi sono aggiornati in modo dilazionato dipendente dalla loro politica amministrativa, ma generalmente sono molto aggiornati anche i mirror.

Nel seguito quando si userà l'indicazione `$TEXMF` ci si riferirà alla cartella o folder che funziona da "radice" per gli "alberi" di cartelle del sistema TEX. Questo sistema, infatti, ha i suoi file collocati in molte cartelle gerarchicamente collegate fra di loro in strutture ad albero; la radice di ciascuno di questi alberi è posta nel disco fisso dell'elaboratore in posizioni diverse a seconda della macchina e del sistema operativo. Ogni utente conosce la sua macchina e sa dove trovare le cartelle radice dei suoi programmi preferiti.

Tuttavia, tranne dove indicato in modo esplicito, qui si userà solo il segno di barra diritta '/' come separatore fra i nomi delle cartelle; questo è il separatore normale per le macchine che lavorano con i sistemi operativi di tipo UNIX, incluso il sistema operativo delle macchine Macintosh recenti, etichettato con la sigla Mac OSX. Nelle macchine Windows il separatore è notoriamente la barra rovescia '\'; ma questo, all'interno dei file che contengono i testi da comporre con i programmi del sistema TEX va sostituito con la barra normale, poiché la barra rovescia per il sistema TEX ha un significato speciale.

I lettori ci vorranno perdonare se abbiamo dato la preferenza sia a TEX Live, sia alla barra diritta; questo è anche legato al fatto che come il sistema TEX fa parte del software libero, così abbiamo dato la preferenza ai sistemi operativi liberi o quasi. Ovviamente non c'è niente di male a servirsi di prodotti commerciali, che talvolta offrono qualcosa di più dei prodotti liberi. Anche nel campo delle distribuzioni del sistema TEX ci sono diverse distribuzioni commerciali che offrono qualcosa in più. Tutti i prodotti commerciali, però, danno qualcosa in meno, nel senso che l'utente è vincolato al prodotto e deve talvolta rinunciare alla possibilità di trasferire agevolmente programmi e documenti da una macchina ad un'altra con un diverso sistema operativo o priva dello stesso prodotto commerciale.

Una certa conoscenza preliminare del sistema TEX è utile per comprendere a fondo questo testo; in ogni caso si dà per scontato che l'utente conosca bene come lavorare sulla propria macchina di elaborazione e abbia una certa familiarità con l'uso della finestra chiamata "Terminale" o "Prompt dei comandi" nei vari sistemi operativi. Talvolta il sistema TEX richiede necessariamente di dare dei comandi in linea, scritti cioè in una di queste finestre specificatamente destinate ad interagire con la macchina mediante comandi scritti, senza usare il mouse o altri comandi "scorciatoia".

Si raccomanda infine di non confondere i tre elementi seguenti

shell editor Uno *shell editor* è uno strumento utilissimo per lavorare con il sistema TEX, ma non è il sistema TEX; è semplicemente un editor di testi che l'utente usa per scrivere il suo testo inframmezzandolo con istruzioni di composizione del linguaggio LATEX; l'editor è poi capace di inviare il file prodotto, detto *file sorgente*, ad un programma interprete del sistema TEX affinché ne esegua la composizione tipografica.

LATEX È un linguaggio di programmazione (non l'unico) per la composizione tipografica di documenti con i programmi del sistema TEX, ma non è il programma di composizione.

Programma di composizione Questo è uno dei vari programmi di composizione facenti parte del sistema TEX che interpreta le istruzioni di composizione inserite dall'utente nel file sorgente del suo documento assieme al testo da comporre; il programma, via via interpreta le istruzioni e acquisisce il testo e li trasforma in testo composto salvando il risultato complessivo in un file di uscita che quasi sempre ha il formato PDF; PDF è l'acronimo che sta per *Portable Document Format*, il formato raccomandato dalle norme ISO per ogni uso successivo del suo contenuto: lettura a schermo, stampa, distribuzione in Internet, archiviazione, eccetera.

Confondere questi tre concetti vuol dire rendersi il lavoro con il sistema TEX molto complicato.

Tuttavia nel seguito si userà la scrittura LATEX per indicare sia il linguaggio sia i programmi che sanno interpretarlo per eseguire la composizione del documento, ma solo quando non è importante distinguere di quale programma si sta parlando. Quando invece è importante distinguere il programma dal linguaggio, allora manterremo la scrittura LATEX per designare il linguaggio, ma useremo esplicitamente i nomi dei programmi scrivendoli con un font particolare: pdflatex, lualatex e xelatex.

I

Volume

Capitolo 1

Composizione sincrona e asincrona

1.1 Introduzione

Come si è detto i word processor sono programmi per la composizione di testi dove le singole parole sono collocate nel testo composto, direttamente visibili sullo schermo; idealmente quanto si vede sullo schermo dovrebbe andare anche in stampa tale e quale, senza modifiche di sorta.

Questo genere di programmi prevede una forte interattività fra compositore e programma di presentazione o di stampa; dato il fatto che ogni tasto produce subito un effetto, così come ogni movimento e click del mouse, il tipo di composizione che viene prodotto viene detto 'composizione sincrona', proprio perché ogni azione del compositore si traduce in una immediata variazione del testo composto.

Va da sé che, perché il programma sia davvero sincrono e il ritardo fra azione e visualizzazione sia trascurabile, la forza del programma deve essere concentrata nella rapidità della presentazione. Tale caratteristica non può che andare a scapito della 'perfezione' della composizione, perché questa dipende da una elaborazione molto più accurata sul testo da comporre. È vero che oggi i programmi di videocomposizione (word processor) sono estremamente rapidi e quindi ogni anno che passa la qualità della loro composizione migliora vistosamente, tuttavia il compromesso fra velocità e qualità esiste sempre.

La 'composizione asincrona' consiste invece nell'introdurre il testo da comporre in un file senza badare al suo aspetto grafico e nel farlo fluire successivamente dentro un programma di impaginazione, dove il compositore può agire (anche in un secondo tempo) per modificare interattivamente il solo aspetto grafico. Durante questo processo può accadere di dover modificare la giustezza di alcune linee o di interi capoversi e quindi può accadere che le linee facenti parte di un capoverso

o l'intero capoverso debbano venire ricomposti e giustificati diverse volte, senza limitarsi a ottimizzare la composizione della parte modificata, ma prendendo in esame l'intero capoverso; questo implica l'ottimizzazione compositiva dell'intero capoverso, non solo una aggiustatina del punto in cui si è eseguita la modifica. Ecco quindi che la composizione avviene in due tempi, l'introduzione del testo e l'ottimizzazione della composizione, avendo però a disposizione l'intero testo da trattare.

Va da sé che la composizione asincrona assicura una migliore qualità di composizione rispetto a quella sincrona, visto che non viene tenuta in nessun conto la velocità di visualizzazione, ma la forza compositiva viene concentrata sulla qualità.

1.2 Il mark-up di LaTeX

LaTeX è un linguaggio di programmazione, detto *mark up*, per certi programmi di composizione asincrona; con un editor di testi l'autore o il compositore inserisce il testo in un file che viene successivamente elaborato con uno dei programmi di composizione del sistema TeX, che agisce da impaginatore; tuttavia il primo file prodotto mediante l'editor di testo non contiene solo il testo in senso stretto, ma contiene una serie di informazioni di mark-up che successivamente permettono al programma di sapere che cosa sta componendo, in modo da eseguirne la composizione secondo le direttive dello stile del documento che si sta elaborando.

Il testo prodotto mediante l'editor di testo è quindi una specie di programma, che contiene sia il testo da comporre, sia i comandi e le altre istruzioni necessarie per il riconoscimento della parte di testo in corso di elaborazione, sia le modalità compositive specifiche che il compositore desidera introdurre.

L'utente non deve spaventarsi con le parole 'linguaggio di programmazione'; per lo più si tratta solo di informazioni di mark-up per specificare titoli, testi speciali, composizione della matematica, e simili. I comandi e le altre specificazioni sono solitamente espresse in inglese (molto semplice, ridotto all'essenziale) evitando quasi sempre le abbreviazioni o gli acronimi.

Un breve esempio consente di capire meglio il genere di difficoltà, o meglio, di facilità di questo tipo di linguaggio di mark-up.

```
\section{Breve esempio}

Il titolo precedente è quello di una sezione;
più propriamente la parola inglese \textit{section} in
tipografia indica ciò che in italiano si chiama
\textit{paragrafo}.

Questo secondo capoverso contiene una equazione numerata:
\begin{equation}
ax^2 + bx + c =0
```

```
\end{equation}
%
Esso contiene anche una tabella centrata:
\begin{center}
  \begin{tabular}{lcl}
    \hline
      Nome      & relazione  & parentela \\
    \hline
      Giovanni  & è          & il papà   \\
      Ada       & è          & la mamma  \\
      Maria     & è          & la figlia \\
      Giuseppe  & è          & il figlio \\
    \hline
  \end{tabular}
\end{center}
Qui c'è la fine del capoverso che contiene
sia una formula sia una tabella.
```

L'insieme di testo e comandi contenuti nell'esempio precedente viene poi composto come nella paginetta della figura 1.1.

Nel capitoli successivi verranno spiegate tutte le informazioni di mark-up e le istruzioni usate nell'esempio. Tuttavia anche il lettore con pochi rudimenti di inglese capisce perfettamente quello che il mark-up ha specificato; le istruzioni per la composizione del materiale tabulare sono un poco più articolate, ma guardando il risultato ci si rende conto che non sono poi così misteriose.

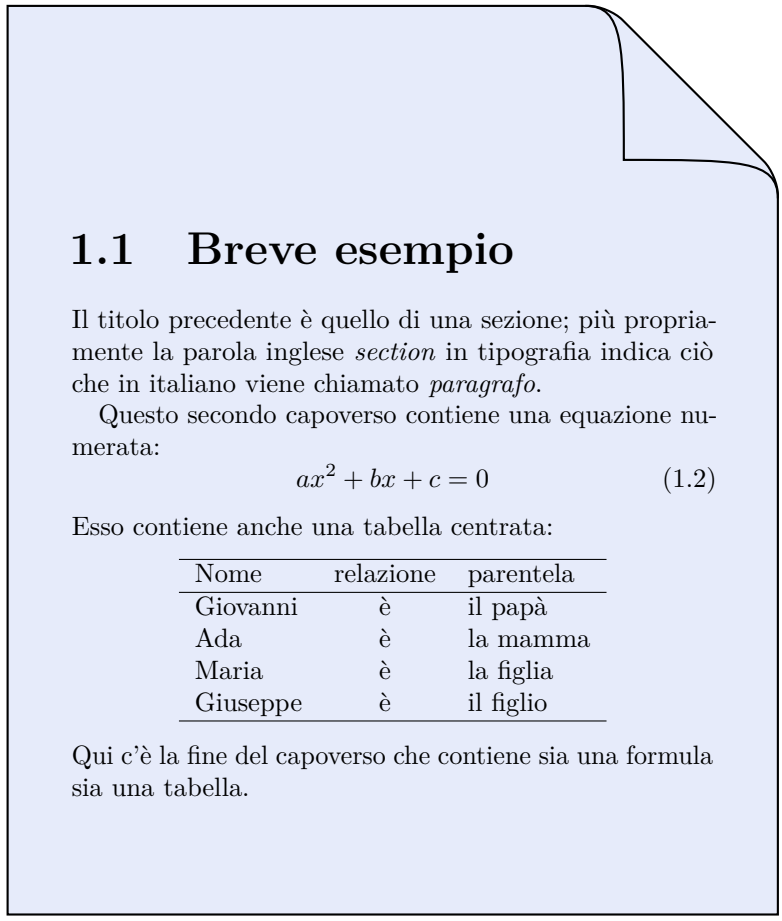
1.3 Scribus: un impaginatore con licenza GPL

Di programmi di impaginazione commerciali ne esistono molti e tutti, essendo destinati al lavoro professionale, sono anche abbastanza costosi. Tuttavia per le tre piattaforme di lavoro principali (Windows, Linux, Mac) esiste l'impaginatore Scribus che, come LaTeX, è rilasciato con licenza libera, in questo caso con la licenza GPL e quindi è *Software Libero*, e chiunque se lo può scaricare dalla rete e installarselo.

Tuttavia merita riportare un piccola parte della documentazione descrittiva di Scribus; qui c'è una libera traduzione del documento che si trova in rete all'indirizzo `http://wiki.scribus.net/index.php/Word_Processing_vs_DTP`.

Prime indicazioni

Molti usano pacchetti di produzione per l'ufficio (*office suite*) come per esempio Microsoft Office™ oppure OpenOffice.org™ oppure LibreOffice™. Ma a un certo punto non sono più soddisfatti delle potenzialità del loro word processor preferito. Forse non possono apportare correzioni fini al layout, oppure alcune loro idee



1.1 Breve esempio

Il titolo precedente è quello di una sezione; più propriamente la parola inglese *section* in tipografia indica ciò che in italiano viene chiamato *paragrafo*.

Questo secondo capoverso contiene una equazione numerata:

$$ax^2 + bx + c = 0 \tag{1.2}$$

Esso contiene anche una tabella centrata:

Nome	relazione	parentela
Giovanni	è	il papà
Ada	è	la mamma
Maria	è	la figlia
Giuseppe	è	il figlio

Qui c'è la fine del capoverso che contiene sia una formula sia una tabella.

Figura 1.1: Semplice esempio di composizione con LaTeX.

non sono realizzabili. Oppure la rivista all'edicola giù all'angolo appare fatta in modo più professionale.

Perciò queste persone si guardano attorno per trovare altri applicativi, e trovano applicativi professionali per il Desk Top Publishing (DTP) come QuarkXPress, InDesign o Scribus. L'ultimo in particolare ha un grande vantaggio: è Free Software, perciò perché non provarlo? Dopo aver scaricato il programma dalla rete, averlo installato e finalmente lanciato, l'autore abituato ai word processor resta seduto davanti allo schermo come "un coniglio di fronte a un serpente". Non sa assolutamente che cosa fare. Tutto appare così strano; e perché non può semplicemente introdurre il testo nel suo documento? Dopo qualche tempo, completamente frustrato, l'autore disinstalla Scribus e torna al suo solito word processor, accettandone le limitazioni e gli svantaggi.

Le principali differenze

- Un applicativo DTP non è fatto per creare testo — almeno se si sta pensando a grandi quantità di testo, cioè molte pagine. Per questo scopo è meglio usare un word processor o un semplice elaboratore di testi anche senza mark-up. Dopo che è stato creato un file di testo, esso può essere importato nell'applicativo DTP. Da questo momento all'interno dell'applicativo DTP si eseguono solo delle piccolissime modifiche al testo e si evitano le correzioni corpose usando lo stesso applicativo DTP. Questo modo di procedere separatamente deriva dalle diverse responsabilità dei professionisti: una persona crea il file di testo, un'altra persona ne crea il layout.
- La maggior parte degli applicativi DTP moderni sono basati sulle scatole (*frame*). Questo significa che tutto il contenuto di una pagina è collocata dentro alcune scatole. "Contenuto" include il testo, le figure, i disegni e ogni altra cosa che l'autore vuol vedere nella pagina stampata. Queste scatole sono spostabili con grande libertà e possono essere collocate in ogni punto della pagina. Il testo di una scatola può esser fatto proseguire in un'altra scatola.
  Si potrebbe obbiettare che anche i word processor hanno le scatole di testo e che è possibile importare una figura o un diagramma nel documento che si sta componendo con il word processor. Tuttavia appena si prova a mettere con precisione al suo posto l'oggetto, lo si vuole ridimensionare, aumentarne o diminuirne la risoluzione grafica, o ruotarlo, si scoprono rapidamente le grosse limitazioni di queste prestazioni "DTP–like". Spesso è difficile impedire che si spostino da sole quando si aggiunge altro materiale al documento.
- Una volta che il testo è nella sua scatola di testo, lo si può manipolare con un altissimo grado di precisione. Un word processor può mettere a disposizione corpi dei caratteri di 6, 7, 8, 9, 10... punti, ma con Scribus la precisione arriva ai decimi di punto. Perché è così importante? In questo modo si può far stare un dato testo dentro un'area di date dimensioni. Per esempio, potrebbe essere necessario adattare la larghezza di una parola con un elemento grafico decorativo che vi giace sotto.
  Si ha la possibilità di allungare un font sia orizzontalmente, sia verticalmente, aggiustare con precisione lo spazio fra le righe (ancora con una precisione di decimi di punto) e lo spazio fra le lettere (accostamenti, crenature, *kerning*). Gli applicativi DTP pertanto danno il controllo completo sulla precisa posizione dei caratteri all'interno del testo, e consentono di adattare il testo ad un dato spazio rettangolare o di qualunque altra forma. Questi sono esempi dove il controllo preciso sulle dimensioni e sulla posizione è assolutamente necessario. Si può adattare il testo in modo che segua il profilo di una figura (o anche il contrario — si provi ad adattare il testo al profilo di una figura con un word processor — pur tuttavia questa potrebbe essere una specifica di layout del DTP), sovrapporre immagini al

testo o viceversa, e creare le separazioni di colore per la stampa.
- Negli applicativi DTP la tavolozza dei colori contiene solo qualche dozzina di colori differenti, chi più, chi meno. Qualunque altro colore si desideri deve essere definito a mano. Questo perché nella stampa professionale in offset, i colori aggiunti possono essere considerati *spot color* e la cosa potrebbe non essere quello che si desidera.
  Il DTP consente di rendere semitrasparenti le immagini, i grafici, il testo — qualunque cosa si possa mettere in una pagina.
- Si possono definire le cosiddette *master page*. Se una pagina normale usa una master page come base, tutti gli elementi già presenti sulla master page appaiono anche nella pagina normale. Si pensi ad una rivista, a una *newsletter*, dove certi elementi, come per esempio le intestazioni, i piedini, i numeri di pagina, i loghi, e il layout della pagina sono ripetitivi, mantenendo però una differenza fra le pagine di sinistra o quelle di destra. All'interno dello stesso documento può essere definita più di una master page; un documento di molte pagine potrebbe fare uso di un numero considerevole di master page.

Che cosa un applicativo DTP non può fare

- Un applicativo DTP serve per creare il layout di un documento; non serve per strutturare un testo. Non può creare riassunti o indici al posto del compositore. Non ci sono funzioni per aggiungere e/o gestire note in calce. Scribus, a differenza di altri DTP, può generare l'indice.
- In generale è difficile connettere un applicativo DTP a dati esterni, come quelli contenuti in un database. Scribus lo può fare mediante un suo plug-in python destinato a ciò.

Aggiungo, su informazione di Gianluca Pignalberi, che Scribus è in grado di elaborare al volo dati da programmi esterni e poi di importarli; fra questi programmi esterni ci sono LaTeX, LilyPond (testi musicali), Gnuplot (diagrammi bi- e tridimensionali) POV-Ray (tracciamento di raggi) e Graphviz (tracciamento di grafi).

Vantaggi degli applicativi DTP

- Controllo assoluto del layout. Gli applicativi DTP non rimescolano mai le scatole per aggiustarle alle necessità del testo o della stampante. Al contrario si devono impostare le specifiche del testo e/o della stampante in modo corretto.
- Il profilo del testo si può adattare a qualunque profilo irregolare.
- Le griglie, i margini e i righelli sono molto utili per collocare correttamente le scatole.

- L'uscita è professionale. I formati PDF o PostScript, o uno di quei formati necessari per quelle enormi macchine di stampa offset? Con un applicativo DTP li si può ottenere tutti.

1.4 Conclusione

Se il lettore di queste note desidera disporre di un potente mezzo per creare dépliant pubblicitari, gli strumenti DTP gli sono indispensabili.

Ma se il lettore vuole produrre testi scritti e illustrati può constatare che LaTeX fa tutto o quasi tutto quello che può fare un applicativo DTP, anche adattare il profilo di un testo a quello di una immagine. Il posizionamento di ogni elemento può essere fatto non con la precisione di un decimo di punto, ma con quella di $1/2^{16}$ di punto. Normalmente non è necessario gestire scatole di testo, ma lo si può fare e in questo testo c'è anche un esempio. Le figure e i grafici sono sempre gestiti come scatole. Il colore può essere gestito sia con una tavolozza ridotta, sia con collezioni di colori abbastanza ampie attraverso un apposito plug-in, il quale consente anche di gestire la trasparenza degli oggetti colorati, testo incluso.

L'uscita può essere di default in formato PDF. La variazione della larghezza e/o dell'altezza del corpo dei caratteri viene gestita automaticamente per la composizione del testo (microgiustificazione), ma LaTeX e i suoi plug-in (estensioni) contengono comandi per modificare, ruotare, e scalare qualunque oggetto; il risultato è ottimale se l'oggetto ha carattere vettoriale.

Di default LaTeX lavora con quattro master page ciascuna delle quali distingue le pagine di sinistra da quelle di destra. Ma mette a disposizione una vasta serie di comandi per creare e poi usare quante master page si vogliano.

Le istruzioni di LaTeX permettono di generare automaticamente gli indici generali e analitici; generano i glossari; sono in grado di produrre diversi livelli di apparati critici nei testi letterari: permettono di comporre la musica; permettono di comporre con qualunque alfabeto e con qualunque tipo di font, anche retrogrado o a ideogrammi; la composizione avviene con i font *bitmapped* e con quelli vettoriali, anche se, ovviamente, ottiene risultati migliori con questi ultimi; quelle istruzioni producono correttamente i riferimenti incrociati e numerano automaticamente tutti gli oggetti numerabili, quali i paragrafi, le figure, le tabelle, le formule, eccetera; sono in grado di includere praticamente qualunque immagine in qualunque formato (be', talvolta con un po' di trasformazioni preliminari) e di produrre disegni al tratto o con sfumature di colore che hanno il pregio di venire composti con le parti testuali formate con gli stessi font usati per il testo. Inoltre, ultimo, ma non meno importante, i programmi di composizione del sistema TeX, fra i quali LaTeX, non ricorrono a formati di file proprietari al fine di poter mantenere l'interoperabilità con i calcolatori più diversi. Il sistema TeX, insomma fa certamente di più di un qualsiasi impaginatore, anche se ha un 'peccato originale', secondo gli utenti che non ne apprezzano abbastanza il valore, cioè: *quello che LaTeX non può fare, essendo un programma di composizione*

asincrona, è quello di consentire al compositore di vedere interattivamente il risultato dei suoi comandi, e questo, come spiegato in precedenza, ha molti vantaggi e alcuni piccoli svantaggi.

Capitolo 2

Nozioni elementari di tipografia

Il gergo della tipografia, come tutti i gerghi professionali, contiene un certo numero di parole che hanno un significato che deriva dalla pratica o dalla tradizione.

2.1 Tipografia e dattilografia

Bisogna innanzi tutto chiarire che comporre tipograficamente è una cosa del tutto diversa dalla composizione con la macchina da scrivere e perciò anche con i moderni sostituti delle macchine da scrivere, costituiti dai word processor.

Questi ultimi oggi sono talmente avanzati che danno l'illusione di poter eseguire una composizione tipografica; ma in realtà, a parte i compromessi fra velocità di presentazione e perfezione della composizione illustrati nel capitolo precedente, il risultato dipende moltissimo dalla professionalità del "tastierista"; costui o costei è la persona che immette il testo e che decide cosa fare, che spazi lasciare, dove mettere il materiale non testuale (per esempio le fotografie) eccetera. Se l'operatore, designato con il vecchio termine di tastierista, non ha abbastanza professionalità, il risultato della composizione è modesto.

La bellezza di una composizione tipografica, a parte il testo, sta nel fatto che la disposizione del materiale da leggere o da consultare non richiama su di sé l'attenzione, ma mantiene quella sobrietà che consente al lettore di recepire il messaggio senza distrazioni inutili e senza zoppicare nel leggere a causa di spaziature irregolari o continui cambiamenti di stile dei caratteri.

Vale la pena di osservare la figura 2.1 dove Herman Zapf, un grande disegnatore di caratteri contemporaneo, usa il suo font Zapfino, destinato proprio alle citazioni o alle epigrafi, per comporre una frase di Oswald Veblen che esalta la matematica (e il suo linguaggio), proprio il motivo che spinse Knuth a "inventare" la tipografia assistita da calcolatore mediante il programma TEX. Una

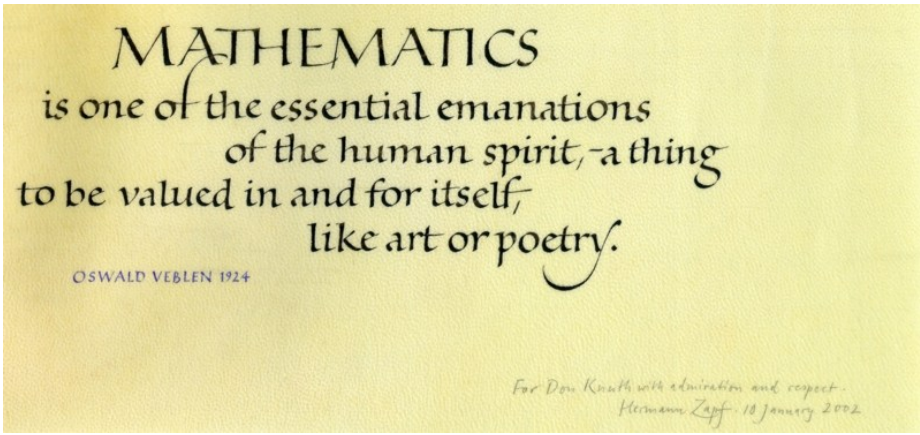


Figura 2.1: Dedica a Donald Knuth

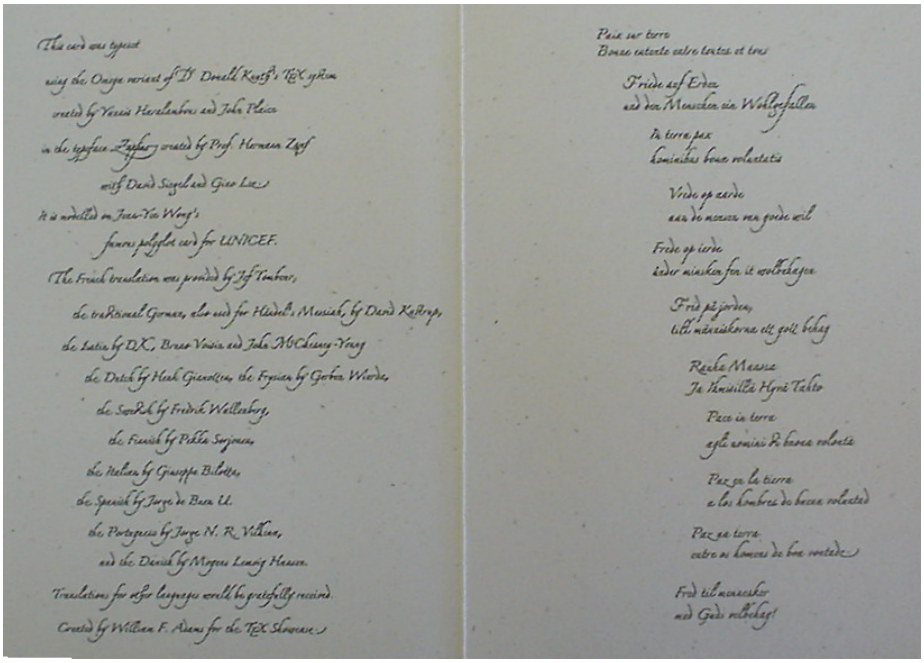


Figura 2.2: Auguri di TUG (TEX Users Group) per il 2003

analoga motivazione celebrativa sta alla base del biglietto di auguri del 2003 della figura 2.2 che l'associazione internazionale degli utenti di TEX ha inviato ai suoi soci.

È chiaro che questa epigrafe e questo biglietto richiamano l'attenzione su di sé; sono fatti apposta! Ma un libro completamente composto con il font Zapfino sarebbe faticosissimo da leggere e il lettore sarebbe continuamente distratto dalle acrobazie calligrafiche eseguibili con questo font[1].

La scelta dei font, la loro grandezza, il loro stile, la loro nerezza, la scelta della distanza fra le righe, i margini, gli spazi bianchi lasciati attorno agli oggetti non testuali, sono tutte cose di competenza del "disegnatore editoriale", non di apprendisti come siamo tutti noi. LATEX, nei limiti di quello che può fare un oggetto "inanimato" come un programma di elaborazione di dati, sostituisce sia il disegnatore editoriale sia il tipografo; non sostituisce il tastierista; se questo pensa di essere davanti ad una macchina da scrivere un po' sofisticata, nonostante tutto quello che LATEX è capace di fare, la composizione risulterà squilibrata e disomogenea, in sostanza, l'opera di un ignorante[2].

La raccomandazione, perciò è la seguente: lasciamo LATEX fare il suo mestiere! Il risultato sarà sicuramente migliore di quello che potremmo ottenere con i nostri maldestri interventi "correttivi".

2.2 Unità di misura tipografiche

In tipografia si usano molte unità di misura tipografiche che spesso, se non sempre, non hanno nulla a che fare con il sistema metrico decimale. Essenzialmente tutto viene misurato in punti tipografici o con i loro multipli generalmente non decimali.

Qui nasce una prima difficoltà interpretativa. Che cosa è un punto tipografico? Ne esistono almeno tre versioni:

1. Il punto tipografico anglosassone definito come la frazione 1/72,27 di un pollice, a sua volta pari a 25,4 mm. Ne segue che il punto anglosassone corrisponde a 0,3514598 mm, poco più di un terzo di millimetro. Questa è l'unità di misura usata dai programmi di composizione del sistema TEX, e ogni informazione metrica che quei programmi forniscono all'utente durante la sua esecuzione è sempre espressa in questo tipo di punti tipografici.

[1] Per caso chi scrive ha trovato il libretto di Lorenza Bonifazi, *. . . e si mandi a tavola – Antiche ricette del Montefeltro*, Raffaelli Editore, Rimini 2006. Il libretto ha la parte contenente le ricette composta con il font Zapfino e per "imparare" a leggerlo ci vuole un pochino. Lo scopo dichiarato di usare questo font calligrafico molto ornato è quello di rendere l'atmosfera originale del quaderno dal quale l'autrice ha ricavato le ricette; il quaderno era stato scritto a mano, con bella calligrafia, da una sua antenata a metà del XIX secolo.

[2] *Ignorante* va preso nel suo significato etimologico: *colui che ignora qualcosa*, nel nostro caso l'arte tipografica. Avrei potuto usare la parola *dilettante*: *colui che si diletta di qualcosa*. Non necessariamente chi si diletta *ignora*, forse all'inizio, ma prosegue per conoscere sempre di più e per dilettarsi meglio. La maggior parte degli utenti di LATEX non sono professionisti, ma dilettanti nel senso più nobile della parola.

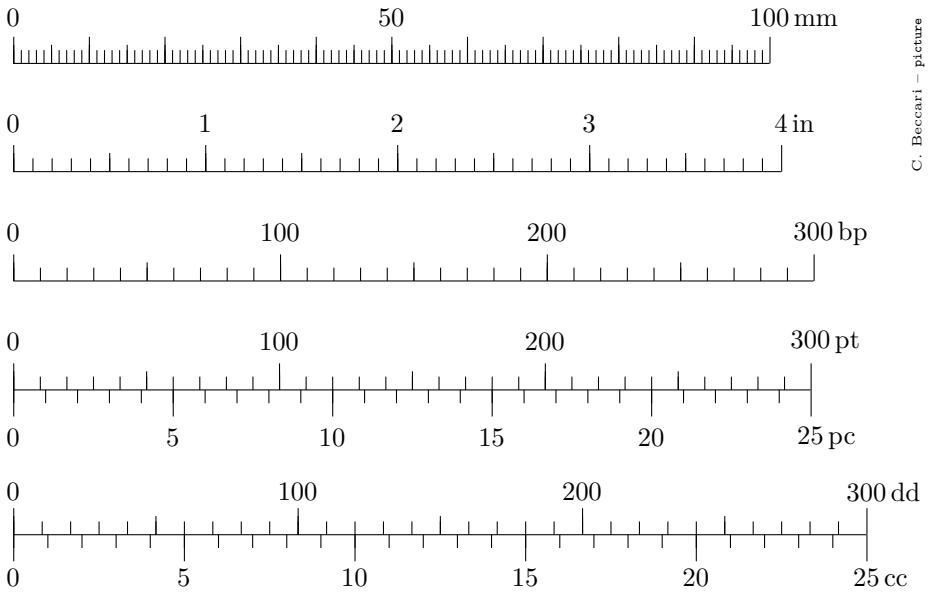


Figura 2.3: Confronto relativo fra le varie scale di lunghezza

2. Il punto PostScript corrisponde a 1/72 di pollice, quindi è un poco più grande del punto anglosassone perché vale 0,35277778 mm. La differenza sembra piccola, ma quando si parla di alcune centinaia di punti la differenza è visibile a occhio nudo.

3. Il punto didot viene usato nell'Europa continentale; esso corrisponde a 0,3760 mm e basta una decina di punti per vedere la differenza ad occhio nudo rispetto al punto anglosassone.

La figura 2.3 presenta un confronto (relativo) delle varie scale metriche che si possono ottenere con le unità di misura descritte in questo paragrafo; se la pagina viene stampata in scala 1 : 1, allora il confronto diventa anche un confronto assoluto.

Non è il caso di preoccuparsi di queste differenze; tuttavia nasce spontanea la domanda: "Ma perché i tipografi non si sono accordati su una unità metrica?" Non so se questa risposta sia adatta: penso che la tipografia sia un'arte praticata in un ambito piuttosto ristretto dove gli interscambi erano fino alla fine dell' '800 abbastanza scarsi, quindi la possibilità di fraintendimenti era modesto. Oggi, nel mondo dell'informatica e della tipografia assistita da calcolatore, sarebbe desiderabile una maggiore uniformità. Infatti oggi si sta imponendo in tipografia il punto anglosassone e in quella assistita dal calcolatore il punto PostScript. È un peccato che il Sistema Internazionale non sia riuscito nel compito di standardizzare una unità di misura adatta alla tipografia (per esempio 'un terzo

Figura 2.4: Caratteristiche dei caratteri mobili. (*a*) Visione d'insieme di un carattere mobile metallico; (*b*) vista laterale del fusto del carattere; (*c*) vista della faccia con il disegno rovesciato del segno da stampare.

di millimetro', ma dotato di tutti i multipli e sottomultipli decimali del caso) come invece, in parte, è riuscito a fare con il carato metrico.

Per indicare lunghezze maggiori in tipografia si usano multipli duodecimali, come il pica, che rappresenta 12 punti anglosassoni, o il cicero che rappresenta 12 punti didot. Bisogna però dire che mentre nel Nord America si usa il pica come unità di misura per indicare la giustezza (lunghezza) della riga di stampa, in Europa e in molte altre parti del mondo più o meno 'decimalizzate', Regno Unito compreso, la giustezza si indica in millimetri. Lo stesso vale per i formati delle carte da stampa; nel Regno Unito sopravvivono denominazioni verbali per i formati dei libri, ma poi la carta è misurata in millimetri. In Italia il cicero è spesso indicato col nome di 'riga', salvo poi a non essere più chiaro se si parla di una riga di testo o di una riga alta 12 punti e, ai giorni nostri, se i 12 punti siano didot o anglosassoni.

2.3 Misure tipografiche

La grandezza dei caratteri è indicata in punti; quando si lavorava con i caratteri mobili metallici, era chiarissimo che cosa volesse dire la parola 'corpo' dei caratteri; questi infatti erano ricavati da blocchetti parallelepipedi di metallo in cui una delle lunghezze rappresentava l'altezza del blocchetto; su una delle facce di base era ricavato in rilievo il disegno del segno da stampare. Si vedano le indicazioni costruite nella figura 2.4, dove sono evidenziate le grandezze dei caratteri mobili.

La faccia di base aveva una altezza e una larghezza. La larghezza era variabile a seconda del segno che il blocchetto conteneva; è evidente che il blocchetto di una 'M' era molto più largo di quello di una 'i'. Invece l'altezza era uguale per tutti i blocchetti dello stesso alfabeto, o meglio dei blocchetti della stessa 'cassa';

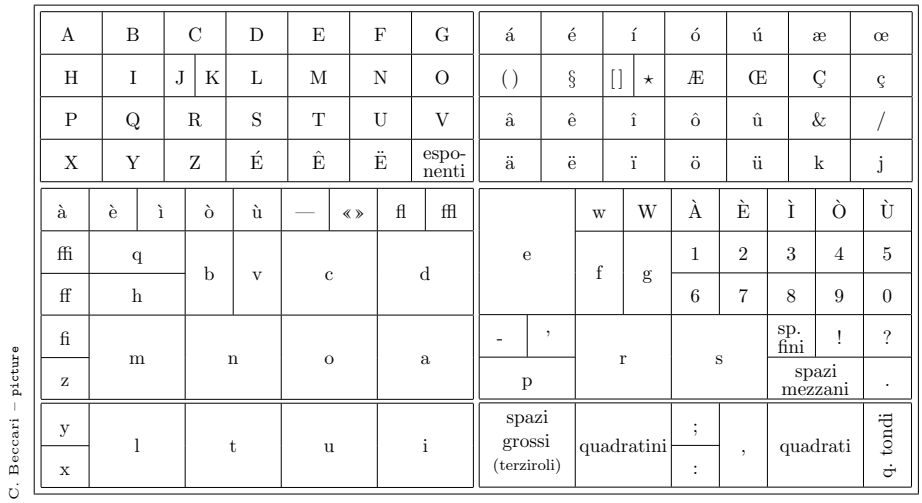


Figura 2.5: Disposizione delle lettere e degli altri segni negli scomparti di una cassa usata in una tipografia italiana

la cassa era un cassetto diviso in tanti scomparti quanti erano i segni minuscoli, maiuscoli, segni di interpunzione, cifre, eccetera, che formavano la collezione dei caratteri dello stesso ‘disegno’; la figura 2.5 mostra la disposizione di una cassa italiana. L’altezza della base di tutti i blocchetti della stessa cassa formava il corpo e veniva misurato in punti.

Vale la pena di ricordare che i nomi inglesi di *lowercase* e *uppercase*, si riferiscono letteralmente alle due metà di una cassa inglese: la ‘cassa inferiore conteneva le lettere minuscole e alcuni altri segni, mentre la cassa superiore conteneva le lettere maiuscole e alcuni altri segni. La disposizione in una cassa italiana, figura 2.5, non ha una separazione così netta, tuttavia anche la cassa italiana, appoggiata su un ripiano obliquo, ha la parte inferiore più vicina all’operatore che contiene le minuscole, mentre la parte superiore, più lontana dall’operatore contiene le maiuscole. Come si vede, dunque, pur avendo casse nazionali diverse, le tradizioni di bottega erano molto simili. Anche in francese si fa riferimento alla cassa bassa e a quella alta per indicare minuscole e maiuscole.

All’indirizzo `http://www.youtube.com/user/EleonoraTallone#p/u/` si trova un breve filmato realizzato da Eleonora Tallone che “intervista” Enrico Tallone, un tipografo artista che compone a mano con caratteri mobili e pubblica testi di grande pregio artistico tipografico. Nel filmato si vedono chiaramente non solo le casse, i caratteri mobili, il compositoio, ma anche altri strumenti tipografici e almeno una macchina da stampa, una platina, in funzione. Un breve filmato semplice ed istruttivo.

Quindi un carattere di 12 pt era un segno qualunque della cassa il cui blocchetto aveva una altezza della faccia di base di 12 punti. Questo evidentemente

non rappresentava l'altezza del carattere, ma, ripeto, l'altezza della faccia del blocchetto. Il fatto che i blocchetti avessero le facce di base della stessa altezza, permetteva al compositore di adagiare tutti quelli di una riga sul compositoio e poi di trasferire la riga completa sulla pagina parzialmente composta con la certezza che tutti i caratteri della riga fossero allineati correttamente.

Inoltre il fatto che tutti i blocchetti avessero i fusti della stessa altezza, come la si vede rappresentata nella figura 2.4(b), assicurava che essi venissero inchiostrati omogeneamente con i rulli inchiostratori e poi prendessero simultaneamente contatto con la carta da stampare e ricevessero la stessa pressione dal torchio di stampa, così da imprimere i segni inchiostrati nella stessa maniera al fine di assicurare una nerezza omogenea dello stampato.

Oggi che i caratteri mobili metallici non si usano quasi più, che significato ha il corpo? Indica solamente l'altezza complessiva della riga di testo, spazio bianco incluso; e in pratica la distanza fra le righe di base di due righe successive quando queste non siano interlineate (nemmeno sterlineate; per questo concetto si veda il capitolo 18).

Ecco allora alcuni nuovi termini: *linea di base*, *interlinea* e *scartamento* o *avanzamento di riga* (dall'inglese *baseline skip*).

La linea di base è quella linea ideale sopra la quale sembrano appoggiati tutti i segni di una riga; sotto la riga di base sporgono le parti discendenti dei segni come quelli delle lettere g, p, q, y; sopra la linea di base sporgono tutte le lettere, ma alcune come b, d, f, eccetera, sono più alte delle altre lettere minuscole, come le lettere a, c, e, m, n, x, eccetera. Siccome tutte queste minuscole sono alte come una lettera 'x', il loro 'occhio' (la loro altezza) in LaTeX viene indicato con il nome di *x-height*, cioè 'altezza della x'.

I *discendenti* sono i tratti delle lettere che sporgono sotto la linea di base e determinano la *profondità* della riga. Gli *ascendenti* sono i tratti delle lettere minuscole che sporgono sopra la linea tangente alla 'x'. Il più alto ascendente è spesso quello dalla lettera 'h'; talvolta sporge sopra l'altezza delle maiuscole, talvolta è alto quanto le maiuscole. In ogni caso l'altezza di ogni riga è determinata dall'ascendente più alto. Non necessariamente il corpo è uguale alla somma della profondità e dell'altezza di una riga di testo.

Come già detto, di fatto il corpo è la distanza di due righe di base composte senza inserire alcuna interlinea. Poiché non necessariamente una data linea contiene tutti i caratteri con i discendenti e tutti quelli con gli ascendenti, il corpo può sembrare maggiore dell'altezza complessiva di una riga, ma in generale, anche se i segni di due righe successive composte senza interlinea non interferiscono fra di loro, la mancanza di interlinea spesso rende il blocco di testo troppo compatto e faticoso da leggere.

Per questo motivo si inserisce una interlinea di un paio di punti, o proporzionalmente di più per i corpi più grandi. Tradizionalmente l'interlinea era quella striscia di metallo che veniva inserita fra due righe successive per distanziarle un poco; in inglese sia chiama *leading* o *lead*, da pronunciare come 'lead' (piombo), perché queste strisce metalliche erano di piombo (o della stessa lega di piombo usata per i caratteri da stampa).

Di conseguenza lo *scartamento* o *avanzamento di riga* rappresenta la distanza effettiva fra le linee di base di due righe successive composte interponendo l'interlinea. Si dirà quindi che un certo testo è composto in corpo 12/14 per dire che i caratteri hanno un corpo di 12 punti e che lo scartamento è di 14 punti, il che implica la presenza di una interlinea di 2 punti. A questo proposito si veda la situazione appena descritta rappresentata nella figura 18.2.

2.4 Le particolarità dei caratteri

Ognuno avrà notato che i caratteri da stampa si dividono sostanzialmente in due grosse categorie, ma in realtà essi sono distinti da una varietà di caratteristiche che permettono moltissime diverse classificazioni.

La differenza fra i caratteri usati dalle vecchie macchine da scrivere, a spaziatura fissa, e i caratteri da stampa vera e propria, a spaziatura variabile, credo che sia evidente a tutti. Oggi i caratteri a spaziatura fissa sono quasi del tutto scomparsi, ma trovano importanti applicazioni, specialmente di carattere informatico, e non solo, quando è necessario evidenziare con il tipo di segno usato certi scritti tecnici, come i brani di linguaggio di programmazione, che non devono essere giustificati e non devono avere nomi o parole chiave divisi in sillabe in fin di riga, ma la cui spaziatura fissa aiuta il lettore a distinguere le variabili dagli operatori, quindi a leggere il codice con più facilità.

I caratteri a spaziatura variabile sono più comodi da usare nei testi giustificati e consentono in generale una lettura più agevole, specialmente se sono caratteri con grazie. Le grazie sono quei piccoli segni, solitamente con andamento orizzontale, alle estremità delle aste che rendono maggiormente definito il contorno dei singoli segni e, specialmente le grazie inferiori, consentono di allineare meglio la direzione della lettura lungo la riga di testo. Quindi una seconda classificazione riguarda la presenza di grazie (*serif* in inglese) o la loro assenza (*sans serif*).

I caratteri non hanno poi la stessa forma; ognuno è in grado di distinguere la forma dei caratteri tondi (*roman* in inglese) dai caratteri *inclinati* (*slanted* in inglese); questi a loro volta si distinguono dal fatto che sono ottenuti dal tondo semplicemente inclinando le aste, oppure sono disegnati apposta con andamento più corsivo (*italic* in inglese), sebbene di questo corsivo sia possibile disporre anche di una versione non inclinata.

Inoltre un alfabeto può contenere sia le lettere maiuscole, sia quelle minuscole dalla forma completamente distinta da quella delle maiuscole, oppure può contenere un alfabeto in cui le lettere minuscole assomigliano ad una versione ridotta delle maiuscole; in quest'ultimo caso si è in presenza del maiuscoletto (*small caps* in inglese).

La nerezza dei segni può assumere livelli diversi; le collezioni di caratteri più ricche hanno diverse gradazioni di nero: chiarissimo, chiaro, normale, neretto, nero, nerissimo. I font per tipografia assistita dal calcolatore spessissimo hanno solo una versione normale e una nera, ma le collezioni più ricche possono avere diverse gradazioni.

Per altro gli stessi segni, con il medesimo grado di nero, possono avere dei disegni ristretti (*condensed*) o allargati (*extended*) rispetto alla dimensione normale.

Una caratteristica che invece non è visibile è la codifica; questo è un concetto tipicamente informatico, nel senso che ogni segno si trova descritto dentro un file insieme agli altri segni dello stesso alfabeto, e per recuperare quel segno dal file è necessario conoscere il suo 'indirizzo'. Esistono moltissimi modi di indirizzare i caratteri (i singoli segni) all'interno di un file e la cosa dipende anche dal numero di segni che il file contiene. Oggi la codifica UNICODE permette di avere file enormi che contengono un numero enorme di segni, disegnati in modo più o meno uniforme anche se appartengono ad alfabeti diversi. Non è raro il caso di file relativi ad una sola collezione che contengono l'alfabeto latino con i suoi segni numerici e di interpunzione, l'alfabeto greco; l'alfabeto cirillico, l'alfabeto ebraico, l'alfabeto arabo, una certa collezione di segni cinesi, giapponesi, coreani, raccolte di segni speciali, come i simboli matematici o i segni astrologici, e chi più ne ha più ne metta, e non è difficile arrivare a qualche decina di migliaia di segni.

La codifica è importantissima, perché si scrive assumendo una certa codifica e si vede sullo schermo il testo correttamente rappresentato; ma poi se si cambia font, si rischia di non avere più la possibilità di decifrare il messaggio; a me è capitato più di una volta di aprire con Word dei documenti composti con un'altra versione di Word e trovare che i segni originali erano stati sostituiti dai segni matematici; la lettura evidentemente era diventata impossibile.

La cosa più importante per i professionisti che si occupano della grafica editoriale consiste nella scelta dei font; due font di forma tonda diritta, nerezza media, estensione normale, con lo stesso corpo e relativi allo stesso alfabeto possono apparire completamente diversi a seconda del rapporto fra lo spessore delle aste spesse e quelle sottili, a seconda della forma delle grazie, a seconda dell'inclinazione dell'asse ottico (non l'inclinazione delle aste), può apparire completamente diverso e può dare luogo ad una lettura più facile oppure più faticosa, se composta in righe con lo stesso scartamento e della stessa lunghezza.

Questo testo è composto con i caratteri della collezione Latin Modern, con codifica a 256 caratteri del sistema TEX[3]; per lo più viene usata la forma di nerezza media e larghezza normale; il corpo normalmente usato è il corpo 10 composto con un scartamento di 12 punti. Talvolta si usa il corsivo per mettere in evidenza delle parole o per scrivere quelle in inglese. I titolini dei paragrafi sono scritti in tondo diritto, nero esteso in corpo 14,4 con un scartamento di 20

[3]Inizialmente si era usata la collezione European Computer Modern con codifica a 256 caratteri, ma poi si è ripiegato sulla "vecchia" collezione con codifica a 128 caratteri, perché alcune delle persone che hanno collaborato alla stesura si sono trovate con una installazione del sistema TEX... troppo personalizzata, al punto da non essere più compatibile con quella degli altri collaboratori. Sono cose che succedono quando si usa uno strumento troppo potente. Dopo diverse versioni di questo testo si è deciso infine di usare la collezione Latin Modern a 256 caratteri, lasciando al singolo collaboratore la libertà di usare questi font, o di escluderli, ritornando perciò all'uso dei Computer Modern.

punti; lo scartamento usato per questi titolini è difficile da vedere perché quasi tutti si svolgono su una sola riga.

2.5 I contrografismi

Logicamente la pagina stampata contiene i segni che convogliano il messaggio, che vengono collettivamente denominati *grafismi*. I *contrografismi* sono le parti della pagina prive di segni, eventualmente sono solo colorati, ma svolgono una funzione essenziale nel disegno grafico della pagina, specialmente nella qualità della composizione, la cui lettura viene agevolata dal loro uso giudizioso.

Non solo l'interlinea è un contrografismo, ma anche la rientranza dei capoversi, gli spazi sopra e sotto le figure, le formule, i titolini; i quattro margini di ciascuna pagina, quello superiore, quello esterno, quello inferiore e quello interno.

Questi ultimi, infatti, determinano la posizione della *gabbia* che contiene il testo e talvolta anche la *testatina* e/o il *piedino* (l'intestazione della pagina e la riga di piè di pagina; *header* e *footer* in inglese); se questi due elementi sono vuoti o quasi vuoti essi contribuiscono più alla grandezza dei margini che alla grandezza della gabbia.

I margini, specialmente quelli esterni, secondo un disegno grafico abbastanza tradizionale, quando sono sufficientemente ampi possono accogliere le note marginali; quindi le note possono trovare posto sia in calce alla pagina, sia nei margini; in qualche scritto critico–letterario esse possono essere anche collocate in fondo a ogni capitolo.

Ma i margini possono accogliere anche parte delle figure o le didascalie delle figure; non si tratta quindi di spazio 'sprecato' ma di spazio che oltre a dare un tono alla pagina può svolgere funzioni ausiliarie al testo in modo creativo e spesso gradevole.

2.6 Le parti di alcuni documenti a stampa

Un libro, generalmente è la forma di stampato più complessa e a questa mi riferirò nel descrivere le parti.

Un rapporto è uno stampato che può assumere la forma di un libretto, ma generalmente ha meno pretese stilistiche, sia per la relativa brevità sia per il contenuto spesso assai tecnico (il che non vuol dire solo ingegneristico, ma anche giuridico, economico, eccetera).

L'articolo è invece un tipo di stampato, di solito assai breve e molto conciso; la concisione implica che non viene 'sprecata' un'intera pagina per presentare il titolo; spesso l'articolo è scritto su due colonne, il che è un artificio per usare gabbie di testo più larghe, capaci di contenere colonne con righe relativamente corte e facili da leggere, mentre se fosse composto con una sola colonna in una gabbia così larga le righe risulterebbero troppo lunghe e quindi faticose da leggere.

Materiale iniziale	
Occhiello	Elemento facoltativo presente prima del frontespizio
Frontespizio	Pagina con i nomi degli autori, il titolo, il nome e/o il logo della casa editrice, e altri elementi facoltativi
Presentazione	Facoltativo: capitolo non numerato contenente un testo di presentazione dell'opera, spesso scritto da una persona diversa dall'autore
Indice generale	In certi libri poco strutturati questo indice può comparire nel materiale finale
Prefazione	Facoltativo: capitolo non numerato dove l'autore descrive la struttura dell'opera e le motivazioni per scriverla; talvolta aggiunge alla fine anche alcuni brevi ringraziamenti
Materiale principale	
Capitoli	Sequenza di capitoli numerati a loro volta strutturati in paragrafi, sottoparagrafi e simili strutture gerarchiche
Appendici	Facoltative: se le appendici sono più di una vanno numerate e messe qui; se il libro contenesse una sola appendice, questa può essere compresa nel materiale finale
Materiale finale	
Appendice	Facoltativa: una sola appendice non numerata
Bibliografia	Facoltativa: ma manca solo nei romanzi e negli altri tipi di testi di svago. Sempre presente nei testi tecnico-scientifici e nei testi di consultazione
Glossario	Facoltativo: può prendere anche altri nomi come "Nomenclatura", "Acronimi", e simili a seconda de contenuto
Indice analitico	Facoltativo: possono essere presenti anche diversi indici analitici tematici; presente quasi esclusivamente nei testi di consultazione

Tabella 2.1: Componenti testuali di un libro

Un libro è uno stampato abbastanza complesso e può essere fortemente strutturato; alcune sue parti accessorie non hanno una posizione assolutamente prefissata; alcune possono mancare. Comunque vale la pena di osservare la tabella 2.1 per farsi un'idea di insieme, ma qui di seguito viene descritta ciascuna parte che forma la struttura riassunta nella tabella.

Un libro può essere confezionato *incassato* (con copertina rigida) o *brossurato* (con copertina morbida incollata); all'interno della copertina esso comincia con un foglio generalmente privo di ogni scritto, seguito da una pagina, sulla cui facciata di destra (il *recto*) può essere riportato l'*occhiello*[4] (o *half title* in inglese);

[4]La parola *occhiello* in italiano ha diversi significati; in particolare nella stampa dei giornali

il retro (il *verso*) è sempre senza testo; la pagina successiva sul recto presenta il frontespizio (*title page* in inglese), mentre sul verso presenta le informazioni di carattere legale richieste, appunto, dalla legge.

La successiva pagina può contenere una dedica sul recto, mentre sul verso è senza testo; se la dedica manca, manca l'intero foglio.

Segue poi sul recto del foglio successivo una sezione non numerata generalmente intitolata 'Presentazione' o con un titolo equivalente. Spesso non supera una pagina, ma può anche occupare qualche pagina.

La successiva sezione, che si apre sul recto, contiene l'indice generale a cui possono seguire, aprendosi sul recto delle pagine, l'elenco delle figure, l'elenco delle tavole, l'elenco delle illustrazioni fuori testo; ovviamente la presenza di questi elenchi dipende dal tipo di libro, dal suo contenuto. Nei libri di carattere letterario, come i romanzi, spesso l'indice si trova in fondo al libro e in Italia era questa la consuetudine di ogni tipo di libro molte decine di anni fa. Oggi molti romanzi e tutti i libri con indici molto strutturati presentano l'indice nelle prime pagine, come descritto sopra.

L'Introduzione segue queste parti iniziali, ma in Italia il materiale iniziale, *front matter* in inglese, termina solitamente prima dell'Introduzione.

Quando si componeva a mano o con le macchine Linotype o Monotype della prima parte del secolo scorso, era tradizionale numerare le pagine della parte iniziale con numeri romani; questo era dovuto al fatto che la parte iniziale veniva composta dopo aver completato e rivisto le bozze del resto del libro. operando in questo modo il numero di pagine della parte iniziale non sarebbe stato conosciuto e quindi era impossibile inserire i numeri di pagina corretti nella parte principale del libro e quindi nemmeno negli indici contenuti nella parte iniziale; perciò, numerando a partire da 1 tutto quanto il materiale dopo la parte iniziale, il problema non si poneva. Oggi anche se si compone in forma elettronica l'intero documento compresa la parte iniziale, magari redatta in un secondo tempo, dopo aver presentato le bozze ai personaggi che avrebbero dovuto presentare l'opera, con LaTeX i riferimenti delle pagine sono tutti corretti dopo aver ricomposto i file sorgente un paio di volte, con una perdita di tempo di poche decine di secondi[5]. Perciò si sconsiglia vivamente la consuetudine di usare la numerazione romana anche se richiama la tradizione tipografica italiana sviluppati lungo diversi secoli.

Dopo il materiale iniziale comincia il corpo del testo, *main matter* in inglese.

Il corpo del libro è talvolta diviso in parti, ma è sempre diviso in capitoli; ogni capitolo è spesso diviso in sezioni gerarchiche che in italiano si chiamano paragrafo, sottoparagrafo, sotto-sottoparagrafo, capoverso e sottocapoverso; in inglese esse si chiamano rispettivamente *section, subsection, subsubsection, paragraph, subparagraph*. Si noti la presenza dei falsi amici 'paragrafo' e 'paragraph'!

---

indica il soprattitolo; nei libri indica una facciata che precede il frontespizio e che contiene solo il titolo del volume.

[5] Questa guida di più di 800 pagine e che contiene l'inserimento di molte immagini, richiede un tempo di compilazione di circa mezzo minuto. Questo è un tempo del tutto trascurabile e non giustifica l'uso della numerazione romana.

Il capoverso è quel tratto di testo che comincia (generalmente con una rientranza) con una lettera maiuscola e termina con un 'punto e a capo'. Un capoverso può contenere diversi periodi, ciascuno diviso in diverse frasi, a loro volta... ma qui si sconfinerebbe nella grammatica della lingua. Spesso singoli periodi, o anche solo delle frasi di una certa lunghezza, sono numerati e, come sotto-capoversi, a seconda dello scritto possono chiamarsi versetti, oppure commi. In generale i versetti sono solo numerati, mentre i commi possono avere anche un brevissimo titolo; in questo caso essi cominciano su una nuova riga di testo, ma nei testi di carattere legale il "capoverso" (che allora si chiamerà 'articolo') viene distinto dal fatto che viene inserito un visibile contrografismo e un titolo, come minimo un numero, fra un articolo e il successivo.

Finito il corpo del testo, cioè dopo l'ultimo capitolo, comincia il materiale finale, la *back matter*, la cui numerazione araba prosegue quella del corpo del testo, ma la numerazione dei capitoli, spesso chiamati 'appendici', ricomincia da 1, o meglio da A, visto che si usa una numerazione diversa, per esempio letterale; ma mentre i capitoli e le parti cominciano sempre sul recto delle pagine, nel materiale finale i capitoli possono cominciare anche sul verso.

Il materiale finale contiene la Bibliografia, gli eventuali glossari, le appendici (capitoli numerati con lettere), e termina, se ci sono, con uno o più indici analitici, i quali vengono evidentemente composti per ultimi per evitare che le loro informazioni siano errate a causa delle correzioni eventualmente apportate nel corpo del testo.

L'ultimissima pagina, sul verso, potrebbe contenere il colophon, una pagina, cioè, nella quale sono descritte le particolarità compositive del testo, i materiali usati e simili altre informazioni di carattere tipografico. Vengono poi lasciate un paio di pagine bianche per poter inserire i risguardi posteriori a cui è incollata la parte posteriore della copertina.

Per ovvi motivi economici le varie pagine di un libro sono ottenute piegando un certo numero di volte dei grandi fogli; queste *segnature* contengono evidentemente un numero di facciate pari ad una potenza di 2; ci saranno perciò libri in ottavi, in sedicesimi, in trentaduesimi, raramente con un numero di facciate superiore, perché lo spessore delle segnature produrrebbe problemi.[6] Resta il fatto che, tolti i risguardi, le pagine interne devono essere un multiplo del numero di pagine di una segnatura; al massimo l'ultima segnatura di un libro formato da segnature di 32 pagine, potrà essere una segnatura da 16 pagine o anche di 8 pagine, ma questo aumenta leggermente i costi di produzione. Nel predisporre la produzione di un libro le tipografie generalmente sono più attente al numero di pagine che non al loro contenuto (il contenuto è responsabilità dell'autore e della casa

[6] Con particolari piegature si potrebbero avere anche segnature il cui numero di facciate corrisponde ad una potenza di due moltiplicata per tre; per esempio si potrebbero avere segnature di 12 pagine, di 24 pagine, eccetera; questi tipi di piegature non sono molto frequenti nei libri, ma si trovano usate in fascicoletti propagandistici e altri simili opuscoli. Per capire come eseguire le pieghe, si prenda un foglio A4, e lo si pieghi in tre come una lettera da inserire in una busta lunga; le pieghe, viste di lato devono ricordale una lettera 'Z'; ora si esegua ancora una piega a metà; si otterrà una 'segnatura' di 6 pagine e 12 facciate delle dimensioni di 110 mm per 99 mm.

editrice); ma è bene che anche l'autore conosca questi problemi in modo da sapersi regolare se deve produrre un testo pronto per la stampa, cioè un file PDF da trasmettere alla tipografia che non vi mette mano se non per ottenere dal file le lastre per la stampa delle due facce dei grandi fogli che formeranno ciascuna segnatura.

2.7 Osservazioni finali

È chiaro che queste poche pagine non sono in grado di dare altro che una breve panoramica della terminologia tipografica, ma spero che il lettore, con queste poche conoscenze, sia stimolato ad osservare con più attenzione i libri che ha per mano e cerchi di valutare i vari elementi descritti, sia in relazione ai caratteri, sia in relazione agli spazi e alla suddivisione dello stampato.

I nomi inglesi che ho indicato servono anche per conoscere il significato di molti comandi o istruzioni del linguaggio LaTeX, che usa quelle parole o quelle brevi locuzioni, ridotte ad una sola stringa senza spazi, per indicare esattamente quegli oggetti individuati dai nomi inglesi.

Capitolo 3

Ortografia tipografica

Che cosa si intende per ortografia tipografica? Tutti conoscono il significato della parola *ortografia*: la scrittura corretta. Ma questo significato di solito viene inteso in senso limitativo come la scrittura corretta delle singole parole.

Quando si scrive con un potente word processor o con un programma di composizione, come quelli del sistema TEX, le regole da osservare sono molto più numerose perché molto più numerosi sono i modi di scrivere sia per l'abbondanza dei segni a disposizione, sia per le famiglie, serie, forme, corpi dei caratteri, sia per la posizione dei segni, sia per la disposizione del testo sulle pagine, sia per l'interpunzione; insomma, la tipografia è un'altra cosa.

È ovvio che l'ortografia tipografica, quindi, comprende anche l'ortografia testuale, ma la scrittura tipografica corretta richiede di rispettare anche altre regole.

Se ne è accennato anche in altri capitoli; alcuni degli argomenti verranno, quindi, ripresi e sviluppati con maggiore respiro, cercando di spiegare il perché di certe regole, anche per imparare a deviare coscientemente da quelle esposte o per dirimere la questione delle ambiguità: "in questo caso si può fare così, oppure così"; va bene, ma allora che cosa è meglio fare? Conoscendo i perché si può trovare la risposta a ragion veduta.

3.1 Ortografia testuale

Non c'è bisogno di raccomandare la scrittura corretta di ogni parola del testo; è del tutto ovvio. Ma la pratica insegna che durante l'introduzione del testo si compaiono numerosi errori di battitura, anche se il "tastierista" conosce benissimo l'ortografia. Si suole dare il nome di *refusi* a questo tipo di errori, perché ai tempi della composizione con i caratteri metallici spesso era necessario fondere di nuovo i caratteri di una o più righe al fine di eseguire la correzione.

Oggi ogni programma di elaborazione di testo permette di fare la verifica ortografica; spesso la verifica può essere fatta solo su testo evidenziato in modo

da poter usare regole ortografiche diverse per lingue diverse. Tuttavia è ben noto che questi correttori ortografici permettono di eliminare un gran numero di refusi, ma non permettono di eliminare quei refusi che generano altre parole valide; i famosi *fischi* e *fiaschi*; *corpo* e *copro*; *viene* e *vine*, anche insolite[1]; l'elenco potrebbe andare avanti a piacere. Ne consegue che la rilettura del testo va fatta da una persona possibilmente diversa dal "tastierista", magari un correttore di bozze professionista; in ogni caso la rilettura non va fatta subito, ma bisogna lasciar trascorrere alcuni giorni e anche il "tastierista" può rendersi conto di altri refusi sfuggiti ai precedenti controlli.

3.1.1 Ortografie alternative

Esistono molte parole che ammettono diverse ortografie, per esempio: *obbiettivo* e *obiettivo*; *uguale* e *eguale*; in questi casi è necessario verificare su un buon dizionario della lingua italiana per verificare se per caso non esistano delle preferenze per l'una o l'altra scrittura, se non esistano degli ambiti testuali nei quali sia preferita una scrittura e altri ambiti nei quali sia preferita l'altra.

Ortograficamente in italiano sono soggetti ad oscillazioni i plurali maschili delle parole che terminano in *-co*, in *-go* e in *-io* e delle parole femminili che terminano in *-cia* e *-gia*. Esistono delle regole sia per i nomi maschili sia per quelli femminili, ma queste regole presentano numerose eccezioni; quindi in caso di dubbio è meglio riferirsi a un buon dizionario.

Per i maschili in *-co* e *-go* vale la regola di default che le parole piane mantengono il suono gutturale (e quindi viene inserita una 'h'), mentre le parole sdrucciole perdono la pronuncia gutturale e quindi non prendono nessuna 'h': *chirurgo* produce *chirurghi*, *fico* produce *fichi*, ma *amico* produce *amici*. Invece *psicologo* produce *psicologi*, *elettrico* produce *elettrici*, eccetera.

Per i maschili terminanti in *-io* con la 'i' tonica (quindi *io* non forma un dittongo) non ci sono problemi e la 'o' muta in 'i' e si hanno quindi due 'i' consecutive come terminazione: *zio* produce *zii*; *formicolio* produce *formicolii*.

Ma per i maschili terminanti con il dittongo ascendente *-io*, una volta i plurali venivano eseguiti con due 'i' (*studii*), oppure con una 'i' segnata da un circonflesso (*studî*), oppure con la desinenza *j* (*studj*) e persino *-ij* (*studij*); oggi, si preferisce generalmente scrivere il plurale con una sola 'i', assumendo che la 'vocale' terminale 'i' assorba la 'semivocale' precedente 'i'. Tuttavia si mantengono le due 'i' anche modernamente quando certi plurali si potrebbero confondere: *assassinio* produce *assassinii* perché questa parola non sia confusa con il plurale di *assassino*; nonostante lo spostamento dell'accento anche *desiderio* produce *desiderii* per non confondere con la seconda persona del presente del verbo 'desiderare', *desideri*; se si usasse sistematicamente l'accento sulle parole sdrucciole non ci sarebbe nessun motivo per derogare dalla regola 'normale'. Di nuovo, nel dubbio è meglio consultare un buon dizionario.

[1] La parola *vine* è il plurale di *vina*, nome di uno strumento musicale indiano...

Per i femminili in *-cia* e *-gia*, purché la 'i' non sia tonica, i plurali oggi si compongono quasi senza eccezioni conservando la 'i' semiconsonantica se la sillaba finale è preceduta da una sillaba aperta, mentre al contrario la perdono se essa è preceduta da una sillaba chiusa[2]: *camicia* produce *camicie*, ma *bilancia* produce *bilance*; *valigia* produce *valigie*, ma *cengia* produce *cenge*. A questa regola della sillaba precedente aperta o chiusa sembrano[3] sfuggire le parole terminanti in *-scia* che hanno costantemente il plurale in *-sce*: *ascia* e *asce*; *biscia* e *bisce*; *striscia* e *strisce*. Come sempre la consultazione di un buon dizionario permette di risolvere eventuali eccezioni a questa regola; infatti *ciliegia* ammette entrambi i plurali *ciliegie* e *ciliege*. Il dizionario della lingua italiana di Aldo Gabrielli (Hoepli) indica entrambi i plurali, ma negli esempi usa solo la desinenza *-gie*.

Anche scrivendo in lingue straniere avvengono gli stessi fenomeni, con l'aggravante che esistono circostanze nelle quali in una nazione si usa una ortografia diversa da quella di un'altra nazione. Tipica è la differenza fra l'inglese americano e l'inglese britannico, dove non solo alcune parole omografe hanno significati completamente diversi, ma parole molto comuni hanno ortografie diverse, per esempio: *colour* e *color*; *behaviour* e *behavior*; *to analize* e *to analise*.

In queste circostanze il compositore deve scegliere una delle grafie alternative e usare sempre quella; per il lettore non c'è nulla di più fastidioso del trovare la stessa parola scritta in modi diversi. Ovviamente, se si cita un brano altrui, deve essere rispettata anche la sua ortografia, ma il fatto di avere evidenziato la citazione fa capire al lettore che quelle parole sono scritte da un'altra persona.

3.1.2 La 'd' eufonica, la 'i' prostetica e gli apostrofi

Nel parlare si usano spesso le 'd' eufoniche e le 'i' prostetiche. Nel parlare si pronuncia spesso una 'd' fra una congiunzione o una preposizione terminante in vocale e la parola seguente specialmente quando inizia con la stessa vocale. Tipiche le pronunce di *ad*, *ed* e *od*. Nello scrivere è generalmente accettata la 'd' eufonica per evitare lo 'scontro' di due 'a' o di due 'e': *ad altro scopo*; *ad avere*; *ed essere*. È anche accettata la scrittura di *ad esempio*, negli altri casi la 'd' eufonica dovrebbe essere evitata.

Questa norma vale a maggior ragione per la 'i' prostetica, quella 'i' che viene spesso pronunciata dopo preposizioni che finiscono con una consonante se precedono parole che cominciano con la 's impura': *in iscuola*, *in istrada*. Anche se questa 'i' viene pronunciata, non dovrebbe venire mai scritta; si tollera la locuzione *per iscritto*.

[2]Una sillaba è aperta se termina con una vocale, mentre è chiusa se termina con una consonante.

[3]Sembrano soltanto se si prende la definizione in senso troppo letterale: è vero che *cia* è preceduto da *s*, ma il nesso *sci* si riferisce ad una pronuncia particolare diversa dalla pronuncia di *s* seguito dalla pronuncia di *cia*; in alcuni dialetti italiani o pronunce dialettali, non è infrequente questa pronuncia separata e l'ortografia italiana non è predisposta per renderla in modo semplice: in piemontese si trova, per esempio, *s-centrà*, ma in molte parti di Italia si sente pronunciare in 'italiano' *s-centrato*.

Probabilmente esistono simili variazioni di grafia anche in altre lingue; è quindi molto opportuno conoscere a fondo la lingua in cui si scrive, oppure, citando brani altrui, bisogna essere certi di riprodurli integralmente nella lingua in cui sono scritti (eventualmente una traduzione in nota potrebbe essere molto gradita al lettore).

L'apostrofo viene impiegato abbondantemente in italiano, in francese, in catalano, anche nella variante valenzana, e in retoromanzo per indicare l'elisione di una vocale; in italiano bisogna stare attenti a non confondere l'elisione con il troncamento; sarebbe sbagliatissimo scrivere *un'uomo* dove invece l'articolo indeterminativo maschile subisce solo il troncamento (la semplice caduta della 'o') e di fatto è un errore rarissimo negli scritti a stampa. Invece è piuttosto frequente incontrare espressioni errate come *qual'è*.

Salvo poche locuzioni ormai cristallizzate, l'elisione si manifesta in italiano solo con gli articoli determinativi maschili e femminili e con gli articoli indeterminativi femminili. Ma oggigiorno l'elisione viene eseguita raramente, limitandola agli articoli singolari e anche in questi casi spesso non viene eseguita: si trova quasi sempre *l'equazione* o talvolta, nei testi tecnici, *la equazione*, ma non si incontra mai *l'equazioni*. Sessanta anni fa era comune vedere scritto *gl'italiani*; oggi non si incontra più questa scrittura. Vista la possibilità di eseguire o di non eseguire l'elisione, il compositore deve decidere come comportarsi e poi deve coerentemente seguire la sua decisione; come al solito il lettore apprezza la coerenza.

L'apostrofo viene anche usato per marcare apocopi e aferesi che oggi in italiano si incontrano poco: *un po' scuro*; *'sta volta*. In espressioni da recitare si usa l'apostrofo per indicare gli imperativi di alcuni verbi: *da' qui*; *fa' attenzione*[4]. In passato era molto comune scrivere *e'* per indicare il pronome maschile di terza persona *ei*, ma quest'uso è durato solo nei primi secoli di vita dell'italiano. Oggi è corretto scriverlo solo nelle citazioni di brani tratti da testi antichi.

In tipografia non bisogna confondere l'apostrofo con l'apice; molti font dispongono di due segni distinti; con i font normalmente usati con LaTeX si usa un solo segno nel testo ('), mentre in matematica il segno (′) è diverso, ma ci pensa sempre LaTeX ad usare i segno giusto se nel file sorgente si usa sempre il segno dell'apostrofo.

3.2 Accenti

In italiano gli accenti obbligatori sono da porre solo sull'ultima vocale delle parole *tronche*; una apposita norma UNI regola la questione, ma in buona sostanza tutte le vocali terminali delle parole tronche ricevono l'accento grave, tranne la 'e' che può riceverlo o grave o acuto secondo precise regole ortografiche.

L'accento acuto (´) si pone su *ché* e sui suoi derivati *perché*, *affinché*, eccetera; sui derivati di *tre*, *re* e *fe'* (fede): *ventitré*, *viceré*, *autodafé*, eccetera; su alcune parole come *scimpanzé*, *sé*.

[4]Si noti la differenza: *dà*, terza persona singolare dell'indicativo di *dare*; *da'*, seconda persona singolare dell'imperativo di *dare*.

L'accento grave (`) si pone sulla terza persona singolare del verbo essere *è* e sui suoi derivati come *cioè*; sulle parole acquisite in italiano e derivate prevalentemente dal francese, come *bidè*, *relè*; su nomi di bevande come *caffè*, *tè* (la scrittura *the* è errata); sui derivati di *me* come *ahimè*; sui nomi propri come *Noè*, *Mosè*, *Giosuè*, *Josè* (anche se in spagnolo si scrive *José*, che però è ugualmente corretto), eccetera.

Si fa osservare che gli italiani sono piuttosto sciatti nel distinguere gli accenti grave e acuto; non è raro trovare scritto *perchè*, comprensibile per il fatto che con la tastiera italiana la 'è' si ottiene premendo un solo tasto, ma si trova scritto anche *é*, che non è nemmeno comprensibile perché con la tastiera italiana bisogna premere due tasti. Il compositore e il "tastierista" che vuol rispettare l'ortografia tipografica, oltre che quella grammaticale, deve quindi porre particolare attenzione alla distinzione di questi due accenti.

La norma UNI è piuttosto poco impegnativa, nel senso che dà indicazioni poco restrittive; essa consente per esempio di usare l'accento acuto sulle vocali 'i' e 'u', vocali naturalmente chiuse, per le quali sembrerebbe più adeguato l'uso dell'accento acuto, che specifica il suono chiuso delle vocali 'e' e 'o'. Mentre sulla 'e' la norma è chiara (e nel caso di dubbio è consigliabile rivolgersi ad un buon dizionario), sulla 'o' delle parole tronche il suono è sempre aperto, quindi l'accento obbligatorio sulla 'o' è sempre grave.

Il filologo italiano Gian Luigi Beccaria usa regolarmente gli accenti acuti sulla 'i' e sulla 'u' (*cosí, piú*), cosa che è permessa anche dalla norma UNI, ma con la tastiera italiana è difficile da realizzare; egli usa anche il circonflesso sui plurali dei maschili terminati in *-io* (*principî*).

Ma la norma UNI consente anche gli accenti facoltativi, cioè sulle sillabe interne. In questo caso, se si vuole distinguere *cólto* (istruito) da *còlto* (raccolto) bisogna scegliere l'accento secondo l'apertura della vocale 'o'; è quanto mai opportuno verificare la pronuncia corretta su un buon dizionario moderno, dove, spesso, la pronuncia è descritta mediante i segni dell'alfabeto IPA (International Phonetic Alphabet).

Il filòlogo Migliorini, membro della commissione che curò la redazione della norma UNI in questione, vi fece introdurre la specificazione che non si accéntano mai le parole piane, se non per indicare il suono aperto o chiuso della vocale tònica 'e' oppure 'o'; si pòssono accentare, e si consiglia di farlo, le parole sdrùcciole e bisdrùcciole[5]; il consiglio diventa quasi un imperativo quando vi possa essere confusione con parole omògrafe con pronuncia piana. Seguendo questa indicazione si distinguerà *prìncipi* da *principi*, ma non si scriverà mai *princìpi*; si distinguerà *séguito* da *seguito*, ma non si scriverà mai *seguìto*; a maggior ragione si distinguerà *làvati* (imperativo) da *lavati* (participio) e a nessuno verrebbe in mente di scrìvere *lavàti*. Tuttavia si incóntrano spesso parole piane accentate in violazione dell'indicazione espressa nella norma UNI. Anche il rispetto di questa norma costituisce un aspetto dell'ortografia tipogràfica.

[5] È quello che si è fatto in questo capoverso.

Non era così un tempo. Su un libro stampato da Bodoni nel 1803 ho trovato che l'accentazione dell'italiano era eseguita con le regole del greco: accento grave sulle parole tronche e accento acuto sulle parole sdrucciole o bisdrucciole, indipendentemente dall'apertura della vocale. Ovviamente una citazione di quel testo richiede il rispetto di quel tipo di accentazione; ma il fatto che il testo sia messo in evidenza come citazione, giustifica il fatto di non essere conforme alla norma UNI (che non esisteva nel 1803!).

3.3 Sillabazione

Secondo alcuni sarebbe meglio comporre in modo che nessuna parola sia divisa in sillabe in fin di riga. Probabilmente è vero, ma la necessità di eseguire la cesura dipende dalla lunghezza della riga di testo e dalla lunghezza della parola; sicuramente non fa parte della 'ortografia' tipografica avere diverse righe *consecutive* che terminano con la lineetta di 'a capo'. Se la giustezza è decisamente piccola è meglio comporre in bandiera allineata a sinistra, piuttosto che con la giustificazione da entrambi i lati. È qui che il compositore deve esercitare il suo giudizio per comporre 'giusto'.

Tuttavia una delle cose che salta all'occhio quando si legge un testo con molte cesure, è non solo se queste cesure siano corrette (ci mancherebbe altro!) ma se sono eseguite in modo da rendere confortevole la lettura. È per questo, per esempio, che le regole di sillabazione realizzate da per il sistema TEX per la lingua italiana evitano di dividere i possibili dittonghi, i gruppi di vocali che potrebbero essere tali, ma che, non sapendo leggere e non sapendo dove cade l'accento, il programma di composizione non può trattare con certezza. In questo modo si perdono possibili punti di cesura, ma la lettura resta molto più confortevole. Si pensi ad una parola come *paura*; le lettere *au* in altre parole formano un dittongo discendente, ma qui formano uno iato e la parola, grammaticalmente, si potrebbe dividere in *pa-u-ra*. Il programma divide invece solamente in *pau-ra*. In effetti non sarebbe sbagliato se esso dividesse come dice la grammatica, ma il lettore si troverebbe a disagio se la riga finisse con 'pa-' e la riga successiva iniziasse con 'ura'. Questo genere di problemi avviene più spesso di quanto non si immagini, ma la sillabazione tipografica (e quindi l'ortografia tipografica) differisce dalla sillabazione ordinaria (e quindi dall'ortografia ordinaria) proprio perché tende ad avere anche il lettore, non solo la grammatica, come metro di misura della 'correttezza'.

Bisogna ricordarsi di selezionare la lingua giusta in modo che il programma di composizione possa usare le regole di sillabazione adatte alla lingua di composizione corrente; bisogna ricordarsi di specificare la lingua ogni volta che si compone in una lingua diversa da quella principale; forse il testo in lingua non subirà nessuna cesura in fin di riga, ma non lo si può sapere in anticipo, quindi è meglio provvedere per tempo.

Se componendo in colonne strette, si vuole aggiungere qualche punto di divisione in fase di correzione delle bozze, si può sempre inserire nell'argomento

del comando `\hyphenation` una lista di parole con iati divise tenendone conto; per esempio si potrebbero inserire (con tutte le varianti e sempre che le parole siano effettivamente usate nel testo):

```
\hyphenation{pa-u-ra pa-u-re i-de-a-le i-de-a-li ba-u-le ba-u-li
             cro-a-to cro-a-ti cro-a-ta cro-a-te}
```

Anche le parole scientifiche ottenute dall'agglutinazione di varie radici è bene che siano divise etimologicamente per facilitarne la comprensione da parte del lettore; le norme UNI che regolano la questione sono piuttosto flessibili in questo senso; esse danno le regole generali per la divisione fonetica, ma non proibiscono al compositore di eseguire la divisione etimologica. Se una parola scientifica di questo genere capita sovente nel testo, è meglio inserirla con le sue variazioni nella lista di `\hyphenation`; se si incontra una tale parola sporadicamente può essere più semplice procedere a mano' LaTeX mette a disposizione il comando `\-` per inserire una cesura "dentro" una parola, ma la divide in due monconi solo in quel puto: per esempio se si introducesse questo comando nella parola `macro\-istruzione` questa verrebbe divisa solo in *macro-istruzione*. Invece, lavorando in italiano e introducendo il carattere attivo `"`, `macro"istruzione` si introdurrebbe una cesura nello iato "oi", ma i due monconi della parola resterebbero comunque divisibili consentendo la divisione nei punti seguenti: *ma-cro-istru-zio-ne*.

Il carattere attivo `"` produce questo effetto solo mentre si compone in italiano e solo se prima dell'inizio del documento si sia specificato il comando `\setactivedoublequote`[6]; non è possibile usarlo componendo in altre lingue, per le quali, però, esistono comandi analoghi formati spesso da due segni.

3.4 Punteggiatura

Il problema della punteggiatura si presenta sia nella composizione del testo sia quando si inseriscono parti matematiche; se ne parla diffusamente anche in altri capitoli. Si tenga presente che i segni di punteggiatura non sono solo i quattro o sei segni ai quali siamo soliti dare questi nomi. In tipografia ci sono diversi segni classificati come interpuntori.

1. I segni come la virgola, il punto e virgola, i due punti e il punto, sono definiti come segni di marcatura logica

2. I segni come il punto interrogativo, il punto esclamativo e i puntini di sospensione (usati anche per le omissioni) sono classificati come segni di espressione.

[6] Quando si usa xelatex e si specifica *babel* come pacchetto da usare per la gestione delle lingue, il doppio apice dovrebbe poter essere usato come descritto qui; se invece si gestiscono le lingue con *polyglossia* il doppio apice non è più attivo e non può venire usato per la divisione in sillabe su base etimologica, a meno che specificando la lingua italiana con i comandi propri di *polyglossia*, non si specifichi l'opzione *babelshorthands*. Lo stesso vale se si usa lualatex.

3. I segni come le virgolette alte, le virgolette basse (in entrambi i casi semplici o doppie) sono classificati come segni di enfasi.

4. I delimitatori sono principalmente le parentesi tonde e quadre, ma anche le coppie di virgole.

5. I segni di sezionamento o di opposizione sono la barra obliqua, il trattino o lineetta, il tratto medio e il tratto o lineato lungo.

6. I segni di sezionamento possono essere usati anche come simboli a sé stanti: il segno di paragrafo, §, e il segno di capoverso, ¶.

Direi che quelli più difficili da usare correttamente sono i semplici segni di marcatura logica. Infatti fin dalle scuole elementari siamo stati tutti esposti alla definizione di tali segni come quelli che marcano le pause. Non è vero; la virgola, il punto, i due punti e il punto e virgola servono per marcare le divisioni sintattico-grammaticali del testo secondo regole che nulla hanno a che vedere con la marcatura prosodica. Questa infatti non può essere marcata adeguatamente con i soli quattro segni indicati, e ogni oratore che debba leggere un discorso ha la sua personale maniera di indicare i segni prosodici per modulare le inflessioni della voce, per marcare le pause d'effetto, eccetera. Questo naturalmente non vuol dire che i segni di interpunzione non contribuiscano alle pause, se uno dovesse leggere ad alta voce, ma non generano necessariamente pause brevi, medie o lunghe in corrispondenza alla virgola, al punto e virgola, e al punto; possono invece aiutare a mutare il tono della voce. Tuttavia queste pause e questi mutamenti non sono la funzione principale della punteggiatura, sono solo l'aiuto alla comprensione per il lettore che esegue le pause e i mutamenti in base a quello che egli ha compreso. Dare la definizione dei segni di punteggiatura mediante la durata delle pause significa scambiare la causa con l'effetto.

I segni prosodici servono per leggere ad alta voce un testo. I segni di punteggiatura servono per dare risalto alle sezioni sintattico-grammaticali del testo e servono, in sostanza, ad aiutare il lettore a comprendere lo scritto leggendo con gli occhi, non con la bocca; gli occhi non hanno bisogno di respirare.

La virgola serve in generale come separatore seriale negli elenchi (per quelli in display si veda il paragrafo 7.3), serve per separare certe frasi subordinate dalla frase principale; usata a coppie, serve per marcare gli incisi, può essere usata per spostare l'attenzione del lettore da una parola ad un altra anche a poca distanza. Le coppie di virgole delimitatrici di un inciso possono non essere apparenti, perché se risultano adiacenti ad un segno di interpunzione di maggiore importanza, ne vengono assorbite.

La funzione seriale è evidente: "Chiare, dolci, fresche acque. . . ".

La funzione di separazione di frasi non è altrettanto semplice da illustrare; ci sono frasi subordinate che si pensano sempre come subordinate e quindi separate da una virgola, ed altre che non svolgono sempre lo stesso ruolo, anche se la grammatica le classifica nello stesso modo.

Per esempio l'ipotesi di una costruzione ipotetica è quasi sempre separata con una virgola dalla tesi: "Se arrivo in tempo, posso cenare con voi." Invece una

frase relativa viene separata con una virgola dalla principale solo se ha carattere esplicativo o appositivo, mentre non è separata dalla virgola se ha significato specificativo o restrittivo: "Il paziente che ha manifestato una forte poliuria va trattato con..."; la frase relativa è specificativa nel senso che la sua eliminazione toglierebbe significato a quello che resta. Viceversa una relativa è esplicativa quando la sua eliminazione toglie qualcosa dal discorso ma la sua assenza non preclude la comprensione del testo restante. La professoressa Mortara Garavelli in [46] mostra un bell'esempio di relative che svolgono ruoli diversi: le parole sono le stesse, mentre i periodi differiscono solo per una virgola:

- Non seguo i programmi televisivi che mi sembrano scadenti.
- Non seguo i programmi televisivi, che mi sembrano scadenti.

I due ruoli della frase relativa sono assolutamente evidenti; nel primo caso il soggetto non segue alcuni programmi televisivi, quelli che gli sembrano scadenti; nel secondo caso il soggetto non segue nessun programma televisivo e spiega che non li segue perché tutti i programmi gli sembrano scadenti.

La virgola serve anche per spostare l'attenzione. Poiché nella frase ordinaria italiana i concetti più "pesanti" vengono collocati alla fine, introducendo una interruzione con una virgola, è possibile spostare l'attenzione su un concetto precedente; si osservi ancora la coppia di frasi proposte dalla professoressa Mortara Garavelli:

- poveri ma belli
- poveri, ma belli

Nel primo caso l'attenzione cade su "belli", mentre nel secondo caso cade su "poveri".

Ecco quindi che molte regole delle grammatiche elementari diventano delle costrizioni che impediscono di comunicare correttamente il pensiero in forma scritta.

Forse anche il problema della punteggiatura esterna alla matematica in display potrebbe essere visto alla luce di queste considerazioni. Può darsi che lo schieramento a favore di uno o l'altro dei due approcci comprenda che l'altro approccio è più conforme alla punteggiatura in generale; può darsi, anzi, che entrambi gli schieramenti si orientino verso una interpunzione esterna alla matematica che non sia né la totale assenza di interpunzione, né la presenza di una interpunzione abbondante. Qui non si prende posizione nel modo più assoluto; si vuole semplicemente osservare che l'interpunzione va vista con uno sguardo più ampio di quello che le grammatiche scolastiche hanno postulato, e che noi tutti ci portiamo dietro fin dai tempi delle scuole elementari.

Il punto e virgola svolge o può svolgere funzioni simili a quelli della virgola, ma solitamente serve per separare frasi autonome grammaticalmente; nella sua funzione di serializzazione serve per separare frasi che contengano esplicitamente o implicitamente anche il verbo. Il ruolo di enfasi è ancora più importante con il punto e virgola che con la virgola.

Il punto serve per terminare un periodo; se dopo si va a capo esso segna anche la fine di un capoverso; può servire anche per le abbreviazioni.

I due punti sono un po' particolari: essi servono per introdurre una frase che sia una spiegazione della precedente; comunque costituisce un modo per creare una aspettativa per ciò che segue.

Vale la pena di ricordare che in inglese la virgola, i due punti e il punto si chiamano rispettivamente *comma*, *colon* e *period*. Queste sono le forme rimaste nella lingua inglese delle denominazioni latine che venivano usate nei manuali di oratoria, come quello di Quintiliano. In latino vigeva la scrittura continua, senza nemmeno gli spazi fra le parole. Solo verso la metà del primo millennio si cominciò a scrivere, almeno nelle iscrizioni sulle lapidi, inserendo un punto a mezza riga fra una parola e l'altra. Per i non latinisti è piuttosto difficile leggere iscrizioni latine con la scrittura continua. Lo doveva essere anche per gli antichi, e Quintiliano insegnava a distinguere le locuzioni dalle frasi e dai periodi (rispettivamente in latino: *comma*, *colon* e *periodus*) mediante dei segni di interpunzione; il comma veniva terminato con un punto basso, il colon con un punto medio e il periodo con un punto alto. Questi erano allo stesso tempo segni demarcatori sintattici e segni prosodici, ma Quintiliano insegnava anche che, per enunciare correttamente il discorso, bisognava prima capirlo, poi lo si poteva leggere ad alta voce.

Nei secoli questi tre segni hanno cambiato forma e funzione; nel testo [47] sono descritte queste evoluzioni in tutte le lingue europee principali; si tratta di una lettura molto interessante, anche se può sembrare una lettura piuttosto pesante. I collaboratori della curatrice in varia forma hanno esposto i loro studi, alcuni in un registro da iniziati, altri in un registro più divulgativo, ma tutti in modo interessante.

In italiano la virgola (da *virgula*, diminutivo latino di *verga*, asta), svolge, tra le altre, la funzione del comma. Gli altri segni hanno nomi descrittivi della forma del segno, ma il punto e virgola svolge il ruolo del colon, mentre i due punti sono particolari nel compito di introdurre una nuova frase. In italiano la parola *comma* è rimasta solo ad indicare una sola modesta articolazione del testo, spesso un solo periodo, di un articolo di legge. Oggi il termine *colon* in italiano indica un segmento ritmico di un testo, individuato da pause metriche o logiche (non necessariamente marcate con segni di interpunzione).

Va ancora notato che ci sono alcune cose fortemente vietate nella punteggiatura, come per esempio separare il soggetto dal verbo con una virgola; apparentemente il Manzoni lo faceva, ma ad una seconda lettura si capisce che le sue virgole hanno lo scopo di marcare certe enfasi che solo apparentemente separano il soggetto dal verbo; ecco un criticatissimo esempio tratto dai *Promessi Sposi*:

> Però, di tante belle parole Renzo, non ne credette una: né che il notaio volesse più bene a lui che a' birri, né che prendesse tanto a cuore la sua reputazione [...].

Oggi noi (che non siamo Manzoni) considereremmo questa frase errata e metteremmo la virgola prima di 'Renzo'. In realtà, leggendo meglio, sembra di guardare in faccia il narratore che, arrivato a questo punto del racconto, strizza l'occhio come a dire: "mica stupido il ragazzo!".

In inglese americano è invece comune incontrare questa funzione prosodica di separazione del soggetto dal verbo, ma non sembra un esempio da imitare nemmeno in inglese; il *Chicago manual of style* condanna questo uso in modo esplicito.

Qui non è il caso di riscrivere il Prontuario della professoressa Mortara Garavelli [46], a cui reindirizziamo il lettore. Però si sottolinea l'importanza della punteggiatura nella buona tipografia; interpungere bene componendo tipograficamente può voler dire trasmettere un messaggio piuttosto che un altro, evitando di travisare il senso del discorso rispetto a come l'aveva in mente l'autore.

Per i segni di espressione, il punto interrogativo '?' ed esclamativo '!', talvolta geminati in '?!', non dovrebbero esserci problemi interpretativi; per i puntini di sospensione è chiara la loro funzione; tipograficamente parlando in Italia si sogliono scrivere i puntini di sospensione attaccati alla parola precedente; nella tipografia anglo-americana i puntini si scrivono attaccati se rappresentano una sospensione, mentre si scrivono staccati se rappresentano una omissione. Nella tradizione tipografica italiana i tre puntini restano tre anche alla fine del periodo, nel senso che conglobano anche la funzione di punto fermo.

I demarcatori come le virgolette alte o basse non richiedono spiegazioni particolari; tuttavia, disponendo di entrambi i tipi di segni, si è soliti dedicare le virgolette basse (chiamate anche: caporali, sergenti, virgolette uncinate, virgolette quadrate) per evidenziare parole o locuzioni importanti, ovvero per racchiudere i dialoghi; le virgolette alte (o inglesi) allora si usano per evidenziare parole usate come tali oppure per segnalarne un uso ironico o traslato, comunque inconsueto. Le brevi citazioni nel corpo del testo vanno racchiuse fra virgolette; una breve citazione che ne contenga un'altra richiede l'uso di virgolette diverse: la citazione interna avrà per esempio le virgolette semplici e quella esterna le virgolette doppie; oppure quella esterna le virgolette basse e quella interna le virgolette alte.

I problemi si complicano dal punto di vista della correttezza tipografica quando si devono evidenziare i dialoghi, per esempio, in un romanzo. Ci sono diversi modi di evidenziare i dialoghi e le battute non facenti parte propriamente di un dialogo. Possono essere introdotte da un tratto medio, possono essere racchiuse fra virgolette basse, come detto sopra, possono racchiudere un'altra battuta; possono essere solo pensate ma non dette, eccetera. Le cose si possono complicare molto. Si veda più avanti anche in merito all'uso del tratto.

Come al solito la coerenza è il metro principale per eseguire le scelte tipografiche giuste. I trattini, le virgolette alte o basse, semplici o doppie sono 'codici' per distinguere le battute.

Un modo coerente per usare questi segni è il seguente:

- si usano le virgolette basse doppie aperte e chiuse per racchiudere ciascuna battuta detta;
- si usano i trattini per separare incisi narrativi all'interno di una battuta detta, per esempio: «Sì, è proprio così, – affermò con decisione – ne sono assolutamente sicura.»
- si usano le virgolette alte doppie aperte e chiuse per racchiudere una battuta dentro un'altra battuta, per esempio: «Mi hai detto testualmente "Cattivo!" con un tono troppo aggressivo.»
- si usano le virgolette alte doppie aperte e chiuse per racchiudere una battuta pensata, per esempio: . . . pensai: "Ora qui devo cavarmela da solo."
- si usano le virgolette alte semplici aperte e chiuse per racchiudere una battuta dentro una battuta pensata, per esempio: . . . pensai: "Gianni mi ha detto 'Sì' con poca convinzione."

Chiaramente questo schema è coerente, ma non è l'unico che si possa usare.

A differenza del francese e di altre lingue, oltre che dalla pratica tipografica italiana dei secoli scorsi, le virgolette alte o basse non richiedono mai spazi aggiuntivi al loro interno; quelle di apertura sono precedute da uno spazio, mentre quelle di chiusura sono seguite da uno spazio o da un segno di interpunzione, seguito a sua volta dallo spazio necessario.

Si faccia attenzione nelle citazioni in lingua straniera, dove le virgolette vengono usate come nella lingua d'origine; in alcune nazioni le virgolette sono usate con la concavità verso l'esterno; in alcune nazioni le virgolette di apertura sono della stessa forma delle virgolette alte di chiusura ma sono poste alla base della riga di stampa; in alcune nazioni le virgolette vengono ripetute all'inizio di ogni riga per le citazioni che durino più righe. Come già detto, in francese le virgolette sono distanziate con normali spazi interparola dal loro contenuto, mentre in italiano questo oggi è assolutamente vietato. Nel XIX secolo anche in Italia andava per la maggiore lo stile tipografico francese, per cui anche in Italia le virgolette erano distanziate, così come anche i segni di interpunzione, compresi la virgola e il punto. Oggi sarebbe inaccettabile. Anche in latino ecclesiastico[7] si usa uno stile tipografico del tipo francese, ma con spazi più ridotti rispetto a quelli francesi.

Le parentesi tonde servono per incisi più importanti di quelli che si possono delimitare con una coppia di virgole; anche le parentesi tonde non vogliono spazi al loro interno. Le parentesi quadre servono per inserire delle correzioni, specialmente nelle citazioni; delimitano, insomma, delle parti che nella citazione originale mancavano o erano diverse. Le parentesi tonde e quelle quadre possono avere anche altre funzioni, come quelle di delimitare le numerazioni delle formule o delle citazioni bibliografiche; sarebbe importante per una buona e corretta tipografia che le parentesi di ogni tipo non svolgessero più di due ruoli, per

[7] Il latino ecclesiastico è latino moderno, non medievale; oltre ad avere uno stile francesizzante, usa gli accenti. Questo sembra sia richiesto per consentire ai fedeli e ai religiosi di leggere correttamente e all'unisono indipendentemente dall'accentazione che spontaneamente userebbero in base alla sensibilità ritmica derivata dalla loro lingua madre.

esempio quello di incisi e di numerazione delle formule per le parentesi tonde, e quello delle correzioni e delle citazioni bibliografiche per le quadre. Sarebbe troppo confuso per il lettore distinguere più ruoli affidati alla stessa coppia di segni.

I trattini hanno ruoli diversi a seconda della loro lunghezza. Il trattino breve serve principalmente come indicatore di cesura in fin di riga; può svolgere anche il ruolo di trattino che unisce due parole associate nel loro significato cumulativo, ma tenute distinte per non formare una sola parola: *indo-europeo*. Va notato che quando il trattino unisce due locuzioni a loro volta formate da più di una parola, il trattino viene separato con spazi insecabili dalla prima e dalla seconda locuzione: *Trentino - Alto Adige*.

Il tratto medio '–' (*en dash* in inglese) non deve essere confuso con il segno meno '−'; LaTeX mantiene anche graficamente una lunghezza diversa. Il segno meno si usa solo in matematica; se si deve indicare una quantità negativa espressa in cifre, la si componga in modo matematico, non si cerchi di simulare il segno meno con un tratto medio; "la borsa ha avuto variazioni giornaliere dal 2% al −3,5%". Il tratto medio viene usato specialmente per indicare intervalli numerici senza ricorrere alla composizione matematica che usa altri simboli più precisi; per esempio, nelle bibliografie, specialmente se si tratta di articoli, è opportuno indicare le pagine nelle quali essi si trovano stampati dentro al volume che li raccoglie; si scriverà dunque: "pag. 321–347".

Nei romanzi il tratto medio serve anche in sostituzione delle virgolette, per demarcare i dialoghi. In particolare una battuta generalmente inizia con un tratto medio e uno spazio; se essa finisce con il punto a capo, non ha bisogno di avere anche il tratto di chiusura. Se invece viene interrotto da una parte narrativa, o il testo prosegue sulla stessa riga con una parte narrativa, allora la battuta viene separata da quanto segue con il tratto medio preceduto da uno spazio. Nella letteratura anglosassone questo ruolo del tratto medio viene svolto dal tratto o lineato lungo.

Il tratto lungo (*em dash* in inglese) viene usato poco nella tipografia italiana; principalmente viene usato per indicare delle omissioni di lettere in una parola, per esempio per non svelare il nome di un personaggio si può scrivere: "il signor A—". Talvolta, imitando le tradizioni tipografiche angloamericane, il tratto lungo viene usato al posto del tratto medio.

Il segno di paragrafo '§' e di capoverso '¶' vengono usati raramente; il primo certamente viene usato più del secondo; questo a sua volta trova la sua collocazione specialmente nei libri liturgici insieme ad altri numerosi segni speciali; lo si vede anche nei libri che cercano di imitare la scrittura dei codici.

3.5 Abbreviazioni

Le abbreviazioni sono un altro terreno infido; spesso esse formano degli acronimi; in questo caso solitamente viene usata la lettera iniziale maiuscola (talvolta anche le due lettere iniziali) di ciascuna delle parole che formano il nome dell'ente o

dell'entità di cui si usa l'acronimo. Alcuni acronimi sono diventati parole a sé stanti di cui si è perso nel comune sentire il significato della locuzione originale: per esempio, la parola comune *radar* è in realtà un acronimo, inizialmente scritto in lettere maiuscole RADAR, che sta per la locuzione inglese *RAdio Detection And Ranging*. L'acronimo CORECO sta per COmitato REgionale di COordinamento. L'acronimo ONU sta per Organizzazione delle Nazioni Unite; ovviamente in altre lingue, come l'inglese, l'acronimo corrispondente è UNO. Come fatto qui, gli acronimi sono composti con lettere maiuscole, preferibilmente in minuscole del maiuscoletto, senza punti di abbreviazione interposti.

Le abbreviazioni più comuni sono formate dalla prima sillaba e dalle prime consonanti della seconda sillaba e vengono terminate dal punto di abbreviazione. Raramente le abbreviazioni sono formate solo dalle prime consonanti della sola prima sillaba. In rari casi, al contrario, l'abbreviazione è costituita dalle prime due sillabe della parola da abbreviare seguite dalle prime consonanti della terza sillaba e terminando il tutto con il punto di abbreviazione; per esempio 'teorema' può essere abbreviato in 'teor.'

Il burocratese ci ha inquinato con usi strampalati delle abbreviazioni, duplicando alcune lettere per indicare il plurale dalla parola abbreviata; questo avviene anche negli acronimi: VVF (Vigili del Fuoco), OO.SS. (Organizzazioni Sindacali), pagg. (pagine), sigg. (signori), proff. (professori). Sebbene il meccanismo sia chiaro, il plurale indicato in questo modo è pleonastico e fuorviante, visto che il significato plurale è implicito dal contesto.

In alcuni altri casi si sono sviluppate abbreviazioni stenografiche usando solo alcune consonanti della parola da abbreviare: btg. (battaglione), cfr. (confronta, oppure il latino *confer*), e simili.

Altre volte vengono usate abbreviazioni quando non c'è nessun bisogno di abbreviare; quando si abbrevia 'sig.r' al posto di 'signor' si risparmia solo una battuta; lo stesso avviene per 'sig.ra' e 'sig.na'. Certamente queste abbreviazioni sono più adatte a una lettera commerciale che a un testo a stampa; il burocratese sia lasciato alla burocrazia.

Ma questi malvezzi si trovano anche negli scritti a stampa; per esempio in una bibliografia a piè di pagina può capitare di citare una nota e il numero di un volume; entrambi ammettono l'abbreviazione 'n.', ma è evidente che non si può usare la stessa abbreviazione per cose diverse; quindi è meglio non abbreviare piuttosto che abbreviare 'nota' in 'not.'; piuttosto si abbrevierà 'numero' in 'num.' riservando 'n.' alla nota, ma in entrambi i casi è meglio evitare l'abbreviazione, oppure, se proprio manca spazio, si può usare il simbolo № mediante il comando `\textnumero` disponibile se si usa il pacchetto *`textcomp`* necessario per usare i simboli della collezione di font chiamata *TEX Companion Fonts*.[8]

Direi anche che le abbreviazioni del tipo F$^{\text{lli}}$, Ill$^{\text{mo}}$, Chiar$^{\text{mo}}$ con o senza sottolineatura dell'esponente, dovrebbero essere evitate in ogni modo, anche in burocratese, tranne nel caso in cui si debba scrivere la ragione sociale di un

[8] Dal 2020 tutti i simboli speciali di *`textcomp`* sono definiti nel nucleo di LaTeX e non occorre più invocare questo pacchetto.

ente il cui "logo" comprende proprio questo genere di abbreviazioni. Perché non scrivere semplicemente 'Frat.', 'Ill.', 'Chiar.'?

Si ricorda infine che le abbreviazioni, eccettuati gli appellativi per i quali valga la pena, non si usano mai nel testo corrente, ma si possono usare solo negli incisi fra parentesi; si dirà dunque: "l'equazione 23 dimostra che..." , e "(v. fig. 33)", ma non si dirà "l'eq. 23 dimostra che..." o, peggio ancora, "la 23 dimostra che..."[9].

3.6 Appellativi e maiuscole

In italiano le maiuscole si usano con grande parsimonia; purtroppo l'imitazione dell'inglese ci ha abituato a certe forme dove le maiuscole si sprecano.

Cominciamo dagli appellativi. Questi sono quei nomi che permettono di distinguere le persone o sul piano professionale, o sul piano istituzionale, come ingegnere, professore, re, presidente, eccetera. Anche il semplice 'signore' è un appellativo che la buona educazione fa premettere (nel genere e nel registro giusto) al nome di qualunque persona. Gli appellativi non vanno scritti con l'iniziale maiuscola quando sono seguiti dal nome della persona; quindi diremo semplicemente "l'ingegner Rossi", "il professor Bruni", "il re Gustavo", "il presidente Mattarella". Useremo l'appellativo con la maiuscola quando l'appellativo non è seguito dal nome proprio e si riferisce alla posizione istituzionale: "il Re", "il Presidente", "ho parlato con il Direttore", eccetera. Penso che l'ultimo esempio sia già un po' tirato per i capelli; personalmente scriverei "ho parlato con il direttore". Tuttavia si prenda in considerazione che la Gazzetta Ufficiale generalmente scrive "il Presidente della repubblica" lasciando l'iniziale minuscola a "Repubblica".

In qualche manuale di indicazioni editoriali si trova scritto che i nomi degli enti si scrivono con le iniziali maiuscole se sono locuzioni formate solo da una o due parole; in tutti gli altri casi si scrive con la maiuscola solo la prima parola; apparentemente le istruzioni editoriali della Gazzetta Ufficiale seguono questa regola. Siccome l'articolo fa parte integrante dei titoli di alcuni giornali, bisogna scrivere "La Stampa", "Il Giorno", ma sempre secondo questa regola bisognerebbe scrivere "Il resto del carlino"[10].

Certamente in italiano non vanno scritte con la maiuscola le parole che indicano nomi comuni: figura, tabella equazione, teorema, ..., nemmeno quando vi si fa riferimento con il numero identificativo. Siccome poi la "figura 33" è un oggetto ben determinato, la locuzione "figura 33" deve essere usata con l'articolo

[9]Tuttavia all'interno di scritti molto tecnici, se il numero dell'equazione è racchiuso fra certe parentesi che vengono usate solo per indicare le equazioni, per esempio le parentesi tonde, allora si può omettere del tutto la parola "equazione" e si scriverà "la (23) dimostra che...".

[10]Il carlino era una frazione del baiocco, moneta legale a Bologna nel XIX secolo prima dell'Unità d'Italia. Al caffè si pagava la bevanda alla cassa un carlino ricevendo qualche monetina di resto; con questo resto si comprava il giornale; almeno questa sarebbe la spiegazione del nome del quotidiano bolognese.

determinativo o con la preposizione articolata. In nessun caso introducendo iniziali maiuscole. Così vale per tutti gli altri oggetti numerati o numerabili.

I titoli degli articoli o dei libri hanno in italiano la sola prima parola con l'iniziale maiuscola; i titoli degli articoli vanno sempre citati fra virgolette preferibilmente alte; i titoli dei libri non vanno mai citati fra virgolette ma in corsivo.

L'iniziale maiuscola va usata anche per i nomi propri dei palazzi, dei monumenti, degli aerei, delle navi, dei treni, delle automobili.

Gli aggettivi di nazionalità vanno minuscoli; se sono personalizzati e riferiti ad una persona specifica allora possono avere l'iniziale maiuscola. Parlando degli olandesi si indicherà la parola con l'iniziale minuscola, ma parlando di Van Gogh si potrà dire "l'Olandese", allo stesso modo come quando si dice "il Nostro" o, volendo, "l'Artista". Si tratta di usare un qualche appellativo senza farlo seguire dal nome proprio, ma come detto sopra, è già tirato per i capelli usare l'appellativo maiuscolo quando *non* ci riferisca ad una carica istituzionale.

I nomi dei giorni e dei mesi vanno minuscoli; i nomi dei punti cardinali vanno minuscoli a meno che non vengano usati per indicare delle regioni geografiche; quindi si parlerà "del grande Nord", ma si dirà che "gli uccelli migrarono verso nord".

Per concludere; in italiano le maiuscole si usano con parsimonia e la correttezza tipografica italiana non vuole che si imitino gli anglo-americani, come non si imitano i tedeschi o altri modi di scrivere nazionali che usano le maiuscole con regole diverse dalle nostre. Nel dubbio ci si astenga dall'usare iniziali maiuscole.

3.7 Uso dei font

La scrittura tipografica consente di sfruttare le possibilità espressive dei font mediante l'uso sistematico di certe serie o di certe forme che, scrivendo a mano, non sono disponibili. Non è opportuno fare sfoggio di infiniti font, appartenenti a famiglie, serie, forme e corpi diversi. Bringhurst [11] afferma che si dovrebbe usare solo la forma tonda, quella corsiva e quella maiuscoletta e sempre e soltanto nella serie media; questa affermazione è ripresa da Serra in [58]. Il nero dovrebbe essere sempre evitato. A queste raccomandazioni si ispirano sia il pacchetto *ClassicThesis* sia le classi *suftesi* e *octavo*, e senz'altro questa raccomandazione è validissima per testi di tipo letterario. Per quelli di tipo tecnico-scientifico non si rifugge dal nero per i titoli, anche di sezioni di livello inferiore a quello del capitolo, ma lo si esclude nel corpo del testo.

Così si è fatto anche in questo testo che usa le impostazioni di default per la classe *book*, ma si è cercato di evitare ogni comando che impostasse la serie nera, specialmente quando si sono definiti i comandi che compaiono nel file `MacroGuida.sty` usati per certe composizioni particolari necessarie per questo testo. In particolare, rispetto alle precedenti versioni, si è abbandonato il carattere bastoncino nero per caratterizzare i nomi dei pacchetti, e lo si è sostituito con il carattere obliquo della famiglia a spaziatura variabile *variable typewriter type*

della collezione Latin Modern, facilmente distinguibile dal corrispondente font obliquo della famiglia a spaziatura fissa *typewriter type*.

Per evidenziare qualche cosa si usa preferibilmente il corsivo, mai la sottolineatura, pratica esclusivamente destinata alla scrittura con le vecchie macchine da scrivere e alla scrittura a mano. Per i nomi propri si può usare il maiuscoletto; questo viene fatto quasi sempre nelle bibliografie, come viene fatto anche in questo testo.

Come già detto, i titoli dei libri e delle opere teatrali vengono evidenziati in corsivo, mentre i titoli degli articoli pubblicati in periodici o riviste specializzate vengono evidenziati mettendoli fra virgolette; è solo un esempio (conforme, per altro, con le specificazioni del *Chicago Manual of Style* [3]), perché gli stili per comporre le bibliografie sono tanti; vedi anche [70]. I nomi propri dei palazzi, monumenti, aerei, eccetera, vanno scritti in tondo.

Il corsivo si usa anche per scrivere parole in lingua straniera (latino compreso); se una parola è di origine straniera ma è entrata in modo stabile nel lessico italiano, la si può scrivere in tondo, ma in questo caso la si assoggetta alle regole grammaticali dell'italiano, in particolare la si lascia invariata al plurale: tennis, golf, goal, file, software, eccetera. Se la si vuole mettere al plurale con le regole della lingua di origine, la si deve trattare come parola straniera, e quindi la si deve comporre in corsivo. Si scriverà, quindi, "i curriculum" oppure "i *curricula*", ma non si potrà scrivere "i curricula". D'altra parte perché curriculum dovrebbe poter essere declinato, mentre "lapis", "album", "iter", eccetera, non vengono declinati?

Va da sé che se si vogliono usare i plurali delle lingue di origine, è necessario, oltre all'uso del corsivo, anche la perfetta conoscenza della grammatica della lingua d'origine. Attenzione! Questo è un terreno estremamente sdrucciolevole, specialmente se le parole in questione terminano con vocali presenti anche nelle determinazioni delle parole italiane. I frutti *kaki* sono caratterizzati da un nome giapponese, quindi non si può scrivere, anche italianizzando con la sostituzione della 'k' con la grafia corrispondente italiana: *il caco, i cachi*; infatti la lingua giapponese non modifica le parole al plurale e *kaki* vale tanto per *un kaki* quanto per *tanti kaki*. D'altra parte a nessuno verrebbe in mente di trattare i *kiwi* come i *kaki*; si è mai sentito dire o visto scrivere "questo chivo è molto maturo"?

In matematica la scelta dei font è vincolata da norme molto precise, almeno per la varietà di matematica scritta dai fisici e dai tecnologi e dagli scienziati sperimentali, perché ci sono delle norme ISO e delle norme CNR-UNI che ne prescrivono gli usi corretti. Si ripete qui che le norme UNI in Italia hanno valore di legge; qualunque documento con carattere legale, un contratto, un capitolato, una perizia, perde la sua validità se non vengono rispettate le norme; le conseguenze negative potrebbero essere molto, molto pesanti!

3.8 Le note

Le note di commento al testo possono essere di tre tipi: le note marginali, le note a piè di pagina, le note di fine capitolo o di fine documento.

Le ultime andrebbero evitate con cura; sono di disagevole lettura, qualunque funzione svolgano, perché sono lontane dal testo a cui si riferiscono.

Le note marginali possono costituire dei commenti puntuali che si riferiscono al testo che esse affiancano, purché siano sufficientemente brevi e possano trovare posto davvero vicino al testo a cui si riferiscono. Richiedono la presenza di margini larghi e la capacità di allineare la prima riga della nota alla riga del testo a cui si riferiscono. Vengono usate raramente nei testi tecnico-scientifici, ma nei testi di scienze umane vengono usate abbastanza spesso. Ricordiamoci che le edizioni critiche possono avere diversi apparati di note; se in calce ad ogni pagina ci fossero 5 apparati di note, ogni pagina sarebbe tremendamente complicata. Spostare un apparato di note nel margine può costituire una notevole semplificazione.

La tradizione anglo-americana vorrebbe che per le note a piè di pagina il richiamo sia costituito da un numero o una lettera o un simbolo ad esponente e lo stesso richiamo sia ripetuto in calce prima del testo della nota. Fin qui non c'è nessuna differenza con la tipografia italiana. Ma la stessa tradizione anglo-americana vorrebbe che il richiamo nel testo segua l'eventuale punteggiatura dopo la parola a cui la nota si riferisce; si dovrebbe scrivere "amigdala,[9]" e non "amigdala[9],". Questa raccomandazione o tradizione non sembra avere altra ragione che l'estetica; alla luce di quanto è stato detto sulla funzione dell'interpunzione, apparirebbe più corretto scrivere il richiamo di nota prima del segno di punteggiatura, in quanto la nota si riferisce ad una parola che si trova prima di quel segno di punteggiatura, segno che, appunto, separa una parte sintattico-grammaticale da un'altra.

Tuttavia entrambe le collocazioni del richiamo hanno la loro ragione d'essere; il compositore sceglierà quella che gli sembra più adatta al tipo di documento che scrive, ed avrà cura di rimanere fedele a quella scelta per l'intero documento.

3.9 Conclusioni

Come si vede le regole da seguire in tipografia sono molte e sono molte anche le consuetudini tipografiche alternative che si possono seguire; qualunque compositore dovrebbe seguire come prima regola la coerenza. Per rispettare la coerenza compositiva egli si trova avvantaggiato se usa il linguaggio LaTeX, perché mediante la definizione di opportune macro è in grado di comporre molte cose nello stesso stile; se in un secondo tempo dovesse decidere di cambiare stile, gli basta cambiare la definizione della macro e automaticamente tutto il documento viene ricomposto nel nuovo stile in modo corretto e coerente.

Il linguaggio LaTeX, però, non basta; ci sono diverse situazioni, come quelle relative alla punteggiatura, che vanno usate con coerenza, ma che non possono

essere codificate dentro opportune macro. È utile allora che il compositore, se non gli fosse già imposto uno stile della casa editrice per cui compone, si scrivesse autonomamente un manualetto di stile; bastano poche pagine, dove vengono riassunte tutte le regole scelte e, magari, dove sono indicati anche i nomi delle macro che realizzano tutte o alcune di quelle scelte. In un manualetto del genere si possono anche scrivere alcuni promemoria per la punteggiatura, per l'evidenziazione, per le virgolette, per i plurali, per le citazioni in lingua straniera, eccetera, in modo da non dover consultare un libro alla ricerca della regola che non si ricorda più.

Tempo fa chi scrive contribuì con diversi articoli o lemmi al *Dizionario di ingegneria* pubblicato dalla UTET. Fu un opera colossale, magistralmente coordinata dalla redazione editoriale, che dovette assemblare i contributi di centinaia di persone. La precedente edizione era stata ideata e coordinata da Eligio Perucca, professore di fisica sperimentale al Politecnico di Torino.

Eligio Perucca era stato anche il presidente e coordinatore di diverse commissioni o gruppi di lavoro dell'Ente Italiano di Unificazione (UNI) e si era reso conto della necessità di una simbologia unificata nelle scienze e nelle tecniche. Per il *Dizionario di ingegneria* aveva esaminato con cura i font di cui disponeva la UTET e aveva predisposto un manualetto di regole editoriali di molte pagine, non una cosetta da poco, dove oltre alle regole ortografiche indicate in questo capitolo aveva riportato un interminabile elenco di simboli per indicare virtualmente qualunque grandezza fisica potesse entrare in un dizionario di questo genere. Ai tempi della prima edizione il Sistema Internazionale non era ancora stato approvato; in fisica regnava l'anarchia più generale fra sistemi 'cgs', 'cgsm', 'cgse', 'cgs Gauss', 'mksa razionalizzato', 'mksa non razionalizzato', ed altri ancora. Eligio Perucca aveva dispiegato una varietà di simboli e vi aveva associato una varietà di nomi in modo che il *Dizionario* potesse contemperare senza confusione tutte le grandezze che fino ad allora erano state chiamate con gli stessi nomi e indicate con gli stessi simboli, quando in realtà si trattava di cose diverse. I font usati ricorrevano ad ogni possibile serie e ad ogni possibile forma disponibile nella tipografia in cui sarebbe stato composto il *Dizionario*. Si usavano senza sovrapposizioni i font corsivi con grazie, i font senza grazie, i font gotici rotondi; sia l'alfabeto latino sia quello greco; tutti nelle varianti medie e nere.

Questo manuale editoriale fu usato anche per la seconda edizione; fu preziosissimo sia per gli autori, che allora non disponevano ancora dei PC, sia per i compositori, sia per la redazione. Per chi scrive quell'esperienza fu importantissima; in particolare gli insegnò l'importanza dell'ortografia tipografica. Chissà se è riuscito anche a metterla in pratica.

Capitolo 4

Installare il sistema TEX

L'installazione del sistema TEX può essere molto semplice o relativamente complicata; dipende da quali strumenti si usano e da come l'utente sa usare efficacemente il sistema operativo del proprio PC o laptop.

Si distinguono tre casi:

1. Sistemi operativi Windows da Windows Vista in poi con processore a 32 o a 64 bit;

2. Sistemi operativi Linux con processore a 32 o a 64 bit;

3. Sistema operativo Mac OS X dalla versione 10.5 con processore PowerPC oppure Intel a 32 o a 64 bit. Non più aggiornate, sono disponibili ancora le versioni per sistemi operativi precedenti a quello di versione 10.5.

Qui si parlerà delle distribuzioni freeware; per le distribuzioni commerciali l'utente deve valutare bene il rapporto prestazioni/prezzo in relazione alle proprie esigenze. Probabilmente è preferibile cominciare con una distribuzione gratuita, per poi passare ad una commerciale se e quando si sarà constatata le reale necessità di quelle caratteristiche in più che la distribuzione commerciale offre rispetto a quelle gratuite.

Va notato che l'organizzazione degli utenti di TEX, TEX Users Group (TUG), offre la possibilità di scaricare dal suo sito `www.tug.org` l'immagine del disco TEX Live, che, comunque, viene inviato gratuitamente a tutti i suoi soci; questo disco, dati i tempi per il download, può essere anche ordinato alla sede di TUG e lo si può ottenere praticamente al costo della duplicazione del disco o poco più. Questo disco contiene le distribuzioni per tutti i principali sistemi operativi e per tutte le macchine più diffuse; quello che si esporrà qui di seguito può essere ritrovato in quel disco.

4.1 Installazione su macchine Windows da XP in poi

4.1.1 Installare TEX Live

L'installazione di TEX Live sulle macchine Windows avviene come per le macchine Linux; l'unica differenza che il file di installazione che bisogna scaricare dalla rete o che si trova nel DVD di installazione, è più grande, perché deve provvedere ad installare alcuni linguaggi, come per esempio Perl, di cui le macchine Windows non sono normalmente dotate; a parte questo dettaglio che è del tutto trasparente per l'utente, il resto procede esattamente come per le macchine Linux, a meno che non venga descritta qualche procedura particolare riferita a questa o a quella realizzazione particolare del sistema operativo Linux.

Esiste però una alternativa costituita dallo scaricare dal sito CTAN l'immagine ISO del DVD distribuito ai soci dei vari gruppi nazionali o internazionali; e poi installare direttamente da questa immagine.

Infatti per usare l'immagine del disco come se fosse il DVD vero e proprio, basta disporre della versione 8 del sistema operativo Windows; se si dispone di una versione precedente, da WinXP a Win7, bisogna prima installare un "montatore" di immagine ISO come se fosse il disco vero e proprio; chi scrive installò il programma WinCDEmu scaricandolo dal sito `http://wincdemu.sysprogs.org/`. Quello che viene scaricato è in realtà il pacchetto di setup del programma, lanciato il quale in pochi secondi il sistema operativo diventa in grado di aprire una immagine ISO come se fosse un CD o un DVD vero.

Prima dell'installazione di una nuova versione di TEX Live, avendone già installata una versione precedente, conviene rimuovere quest'ultima prima di installare quella nuova. Nell'installazione su macchine Windows, fra le cartelle del sistema c'è la cartella `installer`, dentro la quale compare anche la procedura uninstl.bat; lanciando questa procedura della finestra "prompt dei comandi", viene eseguita la rimozione della versione esistente di TEX Live così da poter eseguire un nuova installazione pulita; eventuali configurazioni locali o pacchetti e programmi non facenti parte della distribuzione vera e propria, precedentemente installati nell'albero di TEX Live radicato in `texmf-local`, vengono conservati e non vengono cancellati dall'esecuzione della procedura suddetta e restano disponibili anche per la nuova versione.

Supponendo di avere scaricato l'immagine `texlive2016-20160530.iso` cliccando su uno dei link della pagina internet `http://www.tug.org/texlive/acquire-iso.html`, basta eseguire un doppio click sul nome del file appena scaricato; questo non solo apre il file come se fosse un CD, ma parte anche automaticamente il programma di installazione; basta rispondere a poche domande nella finestra interattiva che appare, e il sistema si installa da solo. Va notato però che l'immagine ISO viene aggiornata una volta all'anno; nell'esempio riportato sopra si tratta del file che contiene la distribuzione TEX Live2016 creata il 30 maggio 2016. Tuttavia fra le parti installate esiste anche il programma Perl

tlmgr che permette di aggiornare la distribuzione quando si vuole; si consiglia di farlo settimanalmente o, almeno, ogni 10 giorni. Per usare tlmgr efficientemente basta consultare il suo manuale di istruzioni che si apre dando dalla finestra "prompt dei comandi" il comando:

```
texdoc tlmgr
```

Complessivamente l'installazione dall'immagine ISO richiede un po' meno tempo rispetto all'installazione via Web, ed è molto meno soggetta ai capricci della rete, che talvolta è penosamente lenta[1].

4.1.2 Installare MiKTEX

MiKTEX è una distribuzione esclusivamente destinata alle piattaforme Windows.

La forma più semplice per installare il sistema TEX con la distribuzione MiKTEX su una macchina Windows che usi un sistema operativo da Vista fino ai più recenti, è quella di procurarsi il CD di installazione ProTEXt; detto CD può anche essere scaricato come un grande file compresso da uno dei siti dove si trova tutto il materiale relativo al sistema TEX. Per esempio ci si può riferire al sito

`http://www.tug.org/protext/`

e alle istruzioni che vi sono contenute; cliccando sull'apposito link, si tratta di scaricare un grande file di più di 1 GiB, quindi è necessario disporre di una connessione molto veloce. Esiste anche una immagine da masterizzare su un CD, che può essere usata direttamente dal CD senza installare nulla, ma che consente di provare l'intero sistema.

Estratta l'immagine dal file autoestraente scaricato dal sito, l'installazione parte da sola mediante il solito espediente dell'`Autorun.inf`; in realtà basta cliccare gli opportuni bottoni che attivano tutte le operazioni necessarie alle installazioni di tutte le varie parti del pacchetto. Conviene leggere il file `protext-setup-en.pdf` per capire bene cosa fare per l'installazione.

In verità il disco autoeseguibile ProTEXt è solo un programma di installazione; il sistema TEX che viene installato è MiKTEX (attualmente la versione 2.9), di cui si parlerà diffusamente anche in seguito; il file di istruzioni per l'installazione suggerisce di installare la versione *basic*; in questa guida, indipendentemente dalla distribuzione, si consiglia, invece, di installare *sempre* la distribuzione completa.

Il programma di installazione configura l'installazione completa in modo che sia adatta a comporre in ognuna del centinaio di lingue e loro varianti che oggi è capace di gestire; naturalmente è ragionevole pensare che non verranno usate tutte, sicuramente non nello stesso documento; tuttavia la versione *basic* non

[1] Scaricare dai *mirror* italiani è spesso un'impresa: quello della rete GARR funziona abbastanza bene; dall'Italia settentrionale è possibile e conveniente scaricare da siti stranieri svizzeri, austriaci, francesi o anche olandesi o tedeschi, che permettono velocità maggiori. Naturalmente è possibile scaricare da questi siti stranieri anche dall'Italia meridionale, ma le tratte più lunghe implicano anche un rallentamento delle operazioni.

configura il sistema per comporre documenti italiani, quindi, non fosse che per questo, conviene installare la versione completa.

Si noti che questa installazione si occupa di installare anche TeXstudio, un ottimo *shell editor* interattivo per lavorare con il sistema TEX, e il visualizzatore di file PDF SumatraPDF, che è sincronizzabile con l'editor così da poter passare rapidamente dalla finestra dell'editor al punto corrispondente della finestra del visualizzatore e viceversa. Vengono installati anche programmi accessori quali ghostscript, ghostview, praticamente necessari con ogni installazione.

Come già detto, la distribuzione MiKTEX è specifica per le macchine Windows; per queste macchine esiste anche la distribuzione TEX Live, che è multipiattaforma; essa può essere installata al posto di MiKTEX, ma generalmente gli utenti Windows preferiscono installare la distribuzione MiKTEX, la cui gestione appare loro più semplice. Non si nasconde il fatto che il procedimento di installazione può sembrare più semplice, e sotto molti aspetti lo è, ma i problemi nascono dopo: infatti se si accetta l'impostazione predefinita, viene installata una versione di base, cosa che è sempre sconsigliabile fare, a meno che non si disponga di poco spazio sul disco fisso; questo era un problema con desktop e laptop di altri tempi, perché quelli di oggi dispongono di dischi fissi talmente capaci che sembra impossibile riempirli completamente. Inoltre, come già accennato, la versione di base non è configurata per comporre testi in italiano, per cui l'utente, specialmente se è nuovo a MiKTEX, si trova in difficoltà. Poi talvolta non si riescono a scaricare i pacchetti mancanti dalla rete, non per difetto di MiKTEX, ma per impedimenti causati dalla rete. Insomma, il forum del gruppo italiano GUIT contiene quasi giornalmente domande di aiuto da parte di utenti di MiKTEX che non riescono a configurare correttamente la loro distribuzione. Perciò si prenda in considerazione la possibilità di non installare MiKTEX e si veda nel paragrafo precedente la descrizione della installazione di TEX Live su una macchina Windows.

Terminata l'installazione guidata, il sistema è pronto per l'uso; se si è scelto di installare la versione di base, bisogna configurare il sistema per l'uso anche della lingua italiana.

4.1.3 La visualizzazione dell'estensione dei file

Da molti anni i sistemi operativi Windows usano una funzionalità di sistema per visualizzare le cartelle e i lori contenuti che si trovano nel disco fisso o in altri dispositivi di memorizzazione di massa. Si chiama 'Esplora file' o 'Esplora risorse' o 'Explorer'. Nei sistemi operativi più recenti se ne trova l'icona specifica nella barra in basso nello schermo.

L'impostazione di default per tutti i file è quella di mostrarne solo il nome se l'estensione è già nota al sistema operativo. Questo succede con la maggior parte dei file usato o prodotti dal sistema TEX. Perciò diventa difficile per l'utente distinguere i file solo dal disegnino della sua icona che è piccolissimo se il modo di visualizzare è 'Mostra dettagli'. Bisogna convincere il sistema operativo a

mostrare l'estensione di tutti i file. Siccome i sistemi operativi sono cambiati nel tempo, qui si descrive la procedura per Windows 10.

1. Si apra 'Esplora file'.
2. Nella barra superiore si clicchi il menù 'File'.
3. Nel menù a discesa si clicchi 'Opzioni cartella'.
4. Si scelga 'Visualizzazione'.
5. Si scenda nella lista di possibili proprietà di visualizzazione fino alla riga che contiene la frase "Nascondi le estensioni per i tipi di file conosciuti"; a sinistra di questa frase c'è un quadratino contenente il segno di spunta.
6. Si clicchi sul segno di spunta che dovrebbe sparire.
7. Si clicchi su "Applicare a tutte le cartelle"
8. Si clicchi su 'OK' e si chiuda la finestra di dialogo e la finestra di 'Esplora file'.
9. Si riapra 'Esplora file' che ora dovrebbe mostrare le estensioni di tutti i file.

Questa impostazione è molto importante perché nell'uso pratico del Sistema TEX è frequente dover accedere ai file di sistema direttamente attraverso 'Esplora file'.

I file del sistema TEX che interessano l'utente sono quasi tutti file di puro testo; i sitemi Windows associano il programma Notepad per aprire e leggere il contenuto dei file di testo. Purtroppo questo programma non è in grado di distinguere i codici di fine riga prodotti da programmi che non nascano insieme a Windows, come appunto sono i programmi del sistema TEX; la loro lettura diventa quindi difficile se si cerca di leggerli con Notepad. Quindi è conveniente modificare le impostazioni dei file `.log`, `.aux`, eccetera, in modo che il programma di lettura dei file di puro testo non sia Notepad, ma sia Wordpad. Questa operazione va fatta con le specifiche funzionalità di ogni sistema operativo Windows. Alcuni *shell editor* specifici per l'uso con il sistema TEX dispongono di icone per leggere i file `.log` senza passare attraverso 'Esplora file', ma non per leggere gli altri file di testo, quindi l'impostazione appena descritta è particolarmente raccomandabile. Meglio ancora: si installi l'applicativo Notepad++ che gestisce benissimo i file di puro testo e dispone di tutte le funzionalità necessarie per un editor fi file di puro testo.Si imposti il sistema operativo per usare sempre Notepade++ al posto di Notepad.

4.2 Installazione su Linux

Parlare di Linux come se fosse un sistema operativo unico e omogeneo è un po' azzardato; esistono centinaia di distribuzioni Linux ognuna con le sue particolarità; queste consistono di diversi "package manager", diversi schemi di controllo della validità e compatibilità dei vari moduli, eccetera. Inoltre le dipendenze reciproche dei vari moduli sono molto intricate e spesso diverse da una distribuzione Linux a un'altra.

TEX Live è disponibile anche per i sistemi Linux ma richiede alcune attenzioni, specialmente se la versione Linux su cui si deve installare il sistema TEX è soggetta ai vincoli del consorzio Debian.

In ogni caso quasi tutte le varianti Linux dispongono di loro "repositories" da cui scaricare i loro programmi, tra i quali il sistema TEX. Non se ne parla qui, perché le varianti sono troppo numerose, ma ogni utente Linux sa come usare i suoi "package manager" e quindi scendere nei dettagli è probabilmente eccessivo.

Va però specificato che la versione di TEX Live convalidata dal consorzio Debian è solitamente un po' datata, in passato anche di un paio d'anni, oggi al massimo di un anno, anche perché non è aggiornabile con il programma tlmgr (TEX Live manager). Inoltre costituisce una dipendenza per diversi altri programmi. Esistono due alternative a questi problemi.

1. Si configura il sistema Linux in modo da fargli credere di avere già installato il sistema TEX; come farlo dipende dalla particolare installazione di Linux, quindi ogni utente deve documentarsi in merito facendo riferimento alla sua particolare distribuzione. Dopo questa operazione si può installare TEX Live agendo dal sito CTAN come indicato per il sistema operativo Windows.
2. Si lascia che i programmi che dipendono dalle funzionalità di TEX Live/Debian vi facciano riferimento in modo normale, ma accanto alla installazione conforme alle direttive Debian, ridotta al minimo indispensabile, si installa la versione TEX Live/CTAN completa. Le due versioni possono convivere tranquillamente una accanto all'altra se la seconda versione viene installata seguendo le indicazioni riportate nell'articolo [27][2]. Quindi la prima versione TEX Live/Debian serve per le dipendenze e la seconda versione TEX Live/CTAN serve per usare il sistema TEX sempre aggiornato o aggiornabile a piacere.

In entrambi i casi, sui sistemi Linux sono disponibili diversi *shell editor* adatti per gestire i documenti da comporre con il sistema TEX; Kile è un programma Debian-compatibile e fino a qualche tempo fa funzionava solo con la versione TEX Live/Debian; sembra che recentemente sia in grado di funzionare anche con la versione generale di TEX Live; emacs è nato con Linux stesso, in quanto software libero, ed è quasi sempre già installato o immediatamente installabile con *package manager* di Linux: arricchito con il plugin `auctex` va benissimo per gestire i documenti da comporre con il sistema TEX. Sono però disponibili altri *shell editor* multipiattaforma, da TeXworks a TeXstudio e Texmaker, che sono particolarmente comodi perché sono già sincronizzati con un loro visualizzatore di file PDF interno che permette di eseguire la ricerca diretta e inversa passando dalla finestra di editing al punto corrispondente della finestra PDF e viceversa. emacs fa cose incredibili rispetto a quello che si può fare con questi altri *shell editor*, ma non dispone di un suo visualizzatore interno; non sembra che sia una cosa banale sincronizzarlo con i visualizzatori PDF presenti sulle installazioni

[2]Questo articolo descrive come installare TEX Live sulla versione Ubuntu/Linux, ma contiene anche le piccole varianti per eseguire la stessa installazione sui sistemi Fedora e OpenSuse.

Linux, ma apparentemente gli utenti di questo sistema operativo sono capaci di superare queste difficoltà.

4.3 Installazione su Mac con MacOS X

Da quando esiste il sistema operativo MacOS X e la macchina a sua volta usa un processore Intel Dual Core o superiore, oppure PowerPC, esiste la versione universale di MacTEX, valida per entrambe le macchine dotate di quei processori; per la verità recentemente la distribuzione aggiornata serve per l'installazione solo su macchine dotate di sistema operativo con un numero di versione 10.5 o superiore; per le vecchie macchine con sistemi operativi della serie 10, ma precedenti alla 10.5, si possono scaricare dalla rete le "vecchie" versioni, che però non sono più aggiornate, benché siano ancora disponibili. Senza voler spingere a spese non previste, sarebbe meglio in realtà aggiornare il sistema operativo oppure acquisire una macchina più moderna.

Il file di installazione è[3]:

`http://tug.org/mactex/`

dove seguendo le istruzioni, si può scaricare il pacchetto *MacTeX.pkg* specifico per le macchine Macintosh; scaricato questo pacchetto, basta cliccarci sopra e si apre il solito dialogo di installazione; conviene rispondere che si vuole l'installazione completa. Il programma provvede da solo ad eseguire l'installazione completa di una distribuzione TEX Live adattata al sistema di cartelle dei calcolatori Macintosh.

Essa installa anche gli *shell editor* TeXShop e TeXworks; l'utente che trovi più comodi altri editor deve provvedere separatamente; si veda più avanti a questo proposito. Invece i due programmi citati sopra sono già dotati delle funzionalità necessarie per visualizzare il risultato della composizione e per procedere alla stampa. Infatti, il visualizzatore della composizione è integrato sia in TeXShop sia in TeXworks ed è già configurato per passare puntualmente dal file sorgente al punto corrispondente del file PDF compilato o viceversa; questa funzionalità agevola enormemente la lavorazione del documento che si sta scrivendo.

Viene installata anche l'applicazione TeX Live Utility in `Applications/TeX/`; essa è una interfaccia grafica in stile Macintosh per operare con il TEX Live manager; è molto comodo e intuitivo per eseguire gli aggiornamenti.

Vengono installati anche i programmi BibDesk e LaTeXiT; il primo serve per gestire i database bibliografici; il secondo serve per comporre con pdflatex le

[3]Oggi sovente ci si riferisce ad indirizzi come `mirror.ctan.org/...` invece che ad indirizzi espliciti come quello indicato. Riferirsi ad un mirror è una operazione intelligente, perché si impegna la rete per un tempo minore e generalmente su un numero minore di tratte; il guaio per noi italiani è che, riferendosi alla lista di mirror gestita automaticamente dal software di installazione, spesso si è indirizzati al mirror GARR di Bologna; tuttavia, almeno per gli utenti dell'Italia settentrionale è spesso conveniente riferirsi all'Università Tecnica di Vienna, oppure al mirror olandese UCS, oppure all'INRIA francese. Altrove si è già discusso dei mirror utili per gli utenti italiani.

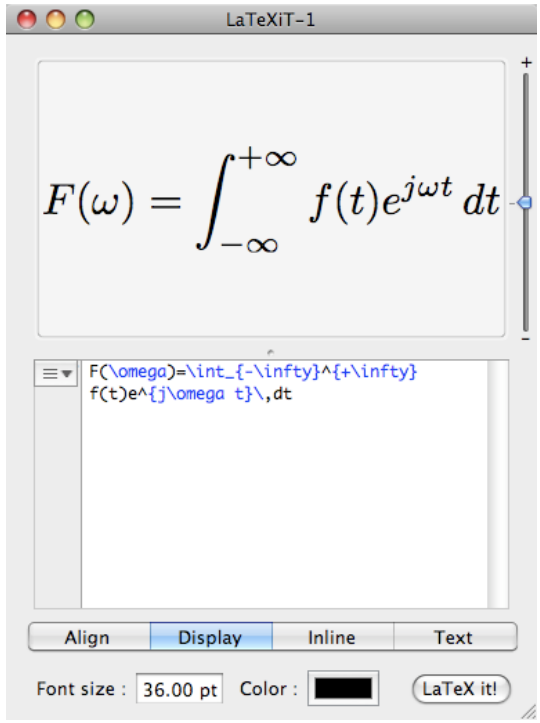


Figura 4.1: Schermata di LaTeXiT

formule matematiche eventualmente utili anche sotto forma di immagine, per inserirle dentro altri formati di documenti. Volendo la si potrebbe introdurre anche nei documenti da produrre con i programmi di composizione del sistema TEX, ma sarebbe veramente stupido inserire una immagine composta tramite LaTeXiT dentro un documento da comporre con uno dei programmi del sistema TEX, quando questo programma sarebbe capace di produrre la formula da solo e senza i problemi di compatibilità del contenuto che si verrebbero a creare.

Al massimo LaTeXiT, a parte gli altri documenti composti con programmi e in formati diversi da quelli che si possono comporre e produrre con i programmi del sistema TEX, può servire come ambiente interattivo per verificare la correttezza della formula via via che la si compone; poi finita la composizione, si copia il codice sorgente e lo si incolla nel file `.tex` del documento da comporre.

Se questo discorso può apparire confuso, si osservi la figura 4.1; essa mostra una finestra di lavoro del programma LaTeXiT divisa in due parti; nella parte inferiore si scrive la formula in linguaggio LATEX standard, senza cioè estensioni di nessun genere, semplicemente usando la tastiera, non il mouse; nella parte superiore, dopo aver cliccato sul bottone in basso a destra, etichettato con il nome LaTeX iT! appaiono i messaggi di errore, casomai ne fossero presenti, oppure l'immagine della formula composta. Se la si vuole salvare come immagine, nella

barra del menù a discesa di File compare la dizione Export image scegliendo la quale si salva il disegno della formula nel formato vettoriale PDF con il nome che gli si specifica.

Lavorando con il sistema TEX, conviene invece selezionare dalla finestra inferiore il codice, copiarlo e incollarlo nel file `.tex` del documento che si sta componendo. Se si vogliono eseguire modifiche per usare comandi non standard o ambienti diversi da quelli standard, basta correggere il codice copiato. Siccome scrivere direttamente in linguaggio LATEX è una cosa che diventa via via più facile man mano che si acquisisce esperienza, ci si accorge in breve tempo che LaTeXiT non è più necessario (quando si compone usando uno dei programmi del sistema TEX) e non lo si usa più o lo si usa solo saltuariamente. Per questo motivo non se ne parlerà più nel seguito.

4.4 Gli alberi di cartelle del sistema TEX

L'installazione dell'uno o dell'altro sistema crea sul disco diverse strutture di cartelle; ovviamente queste strutture di cartelle, gli *alberi* di cartelle, possono essere un po' diversi da sistema operativo a sistema operativo, da installazione a installazione.

La differenza più ovvia è che nella descrizione del percorso sulle macchine Windows il separatore fra i nomi delle cartelle è una barra rovescia, \, mentre sulle macchine Linux e Mac OS X il separatore è una barra semplice, /. In questo testo si userà sempre la barra semplice per due motivi: (*a*) la barra rovescia ha un significato speciale per il linguaggio LATEX, per cui limitarne l'uso alla sua funzione speciale aiuta anche a evitare errori nello scrivere il testo; (*b*) per lo stesso motivo scrivendo in linguaggio LATEX, anche su una macchina Windows, i percorsi dei file devono essere sempre indicati con la barra semplice invece che con la barra rovescia.

Le differenze più importanti riguardano l'installazione TEX Live rispetto a quella MiKTEX. Se ne parlerà più diffusamente più avanti. Qui ci si limiterà ad esporre le similitudini.

Tuttavia l'argomento può sembrare un po' ostico; in un certo senso lo è, ma il lettore non si scoraggi: salti direttamente al paragrafo 4.5. Può tornare sopra questo argomento in seguito, quando vorrà approfondirlo.

Tutti gli alberi, a partire dalla loro 'base' o 'radice', devono essere conformi alla *TEX Directory Structure* (TDS), cosicché il sistema TEX possa trovare i file che gli servono sempre nello stesso modo e con un unico suo programma interno di ricerca dei file che si chiama kpsewhich; questo programma può anche venire usato dall'utente, ma di solito egli ne ignora persino la presenza perché, in effetti, esso svolge le sue funzioni dietro le quinte ed è molto raro che l'utente debba servirsene direttamente. La TDS è descritta nel file `tds.pdf` in grande dettaglio; non è necessario che l'utente fin dalla prima installazione si studi approfonditamente questo documento, ma è importante che ne conosca l'esistenza per poterlo consultare quando occorre. Esso viene installato insieme al sistema e può essere

letto aprendo la finestra comandi, console, xterm, terminal, comunque si chiami quella finestra nel particolare sistema operativo in uso, e scrivendoci dentro il comando:

```
texdoc tds
```

Il comando `texdoc` è utilissimo anche per consultare la documentazione di qualsiasi pacchetto; solitamente basta dargli in pasto il nome del pacchetto e questo programmino esegue le ricerche necessarie per trovarne la documentazione e presentarla sullo schermo.

Ciò premesso, ogni utente farebbe bene a costruirsi un suo albero personale, conforme alla TDS, dove conservare le sue macro, i font che egli si crea, i pacchetti che si scrive; insomma tutto quanto gli sia strettamente personale e sia funzionale al funzionamento dei programmi del sistema TEX. Questo albero personale parte dalla cartella `texmf` che si trova nella sua `$HOME`; questo nome simbolico `$HOME` ha un significato diverso in ogni sistema operativo e deve essere diverso dalla 'radice' sulla quale sono innestati gli alberi installati dal programma di installazione. Normalmente questo nome simbolico rappresenta:

- `~/` nei sistemi Linux;
- `~/Library/` nei sistemi MacOS X;
- `C:\Documents and Settings\`⟨*utente*⟩`\Documenti\` con l'installazione di MiKTEX sulle macchine Windows; sulle versioni Windows Vista, Seven, Eight, 10, vale `C:\Users\`⟨*utente*⟩`\Documenti\`;

dove ⟨*utente*⟩ è il nome con il quale il particolare utente si è registrato sulla macchina.

Non si parlerà più di questo albero personale se non riferendosi ad esso con il generico nome di *albero personale* che, appunto, in quanto tale, verosimilmente non avrà l'intera struttura TDS, ma solo, conformemente a questa struttura, le cartelle di cui il particolare utente effettivamente necessita.

Ogni programma eseguibile del sistema TEX sa dove cercare i vari tipi di file di cui ha bisogno; per ogni tipo di file può eseguire la ricerca in diversi alberi, ma sempre con una precedenza: gli alberi prioritari sono quelli dell'utente specifico, poi vengono quelli della macchina specifica, infine quelli generali dell'installazione. Siccome si tratta di alberi che contengono migliaia di file, la base di ogni albero contiene un database dei nomi dei file che di solito deve venire aggiornato ad ogni modifica. Solitamente non vengono modificati a mano gli alberi del sistema TEX, ma l'albero personale è lì apposta per ricevere ogni file personale dell'utente, perciò questo file viene modificato relativamente spesso. Ogni installazione dispone di comandi specifici per l'aggiornamento dei database dei nomi dei file; MiKTEX dispone addirittura di un *wizard* che richiede solo di cliccare su un bottone di una finestra di interfaccia grafica. In ogni caso i programmi che modificano, aggiungono o tolgono file dagli alberi di sistema provvedono da soli ad aggiornare gli specifici database dei nomi dei file. *Perciò l'utente deve*

ricordarsi sempre di aggiornare questo database del suo albero personale, solo se ha installato la distribuzione MiKTEX; la distribuzione TEX Live, su tutte le macchine, non richiede l'aggiornamento del database dei nomi dei file dell'albero personale.

Si noti con attenzione: l'albero personale, proprio perché personale, non viene creato dal programma di installazione della particolare distribuzione che viene installata sulla macchina; se l'utente ne ha bisogno, deve crearselo da solo; gli si consiglia di rispettare le direttive della TDS, magari limitandosi a creare solo quelle parti della struttura TDS che effettivamente gli servono. Le diramazioni particolari che sarebbe bene creare nell'albero personale sono formate dalle cartelle: `fonts/`, `doc/ source/`, `tex/`, e in queste cartelle verrano via via aggiunte altre cartelle (sempre secondo la TDS) man mano che ne sorge la necessità.

4.4.1 Gli alberi di cartelle di TEX Live

Gli alberi che vengono gestiti più o meno direttamente dal sistema TEX e nelle cui cartelle l'utente in generale non può scrivere nulla, se non operando da "amministratore" o da "root" o da "superuser", con una distribuzione TEX Live sono innestati tutti in ⟨*radice*⟩`/texlive/`. Essi sono:

1. ⟨*radice*⟩`/texlive//`⟨*anno*⟩`/texmf-dist` contiene tutti i file che servono per il funzionamento del sistema TEX, tranne gli eseguibili veri e propri che sono conservati in un'altra cartella.

2. ⟨*radice*⟩`/texlive//`⟨*anno*⟩`/texmf/` contiene tutti i file necessari per la configurazione del sistema TEX generati al momento dell'installazione del sistema TEX; sono quindi file che riguardano la configurazione e il funzionamento del sistema TEX ma sono validi per tutti gli utenti della particolare macchina sul cui disco appare questo albero. Dalla distribuzione 2013 questo albero è stato fatto confluire nel precedente.

3. ⟨*radice*⟩`/texlive//`⟨*anno*⟩`/texmf-config` contiene tutti i file di configurazione di default del sistema TEX; in altre parole sono i file che servono per configurare il sistema prima che eventualmente l'utente specifico o l'amministratore della macchina non decidano di impostare una configurazione particolare, non standard.

4. ⟨*radice*⟩`/texlive//`⟨*anno*⟩`/texmf-var` contiene tutti i file di configurazione del sistema TEX generati in seguito a comandi specifici dell'utente amministratore per una configurazione ad hoc dell'installazione sulla specifica macchina.

5. ⟨*radice*⟩`/texlive//texmf-local` contiene tutti i file utili all'esecuzione dei programmi del sistema TEX e che sono specifici della particolare macchina; in particolare esso contiene certi file di configurazione che dipendono solo da quanto è disponibile sulla macchina specifica.

6. ⟨*radice*⟩`/texlive//`⟨*anno*⟩`/bin` contiene tutti i programmi eseguibili del sistema TEX; il percorso che conduce a questa cartella e alle cartelle che esso contiene viene aggiunto automaticamente alla variabile di sistema `PATH` in modo che i programmi contenuti in questo particolare albero possano venire eseguiti sia dall'utente mediante comandi manuali, sia dagli altri programmi del sistema.

7. `$HOME/texlive` contiene un albero con la sua radice che comincia a sua volta con ⟨*anno*⟩ e prosegue con una TDS contenente i file di servizio e di configurazione generati dall'utente specifico (non in qualità di amministratore) e dai programmi di servizio durante l'esecuzione di composizioni per quello specifico utente; tra gli altri vengono conservati qui i file relativi ai font a matrici di punti che il sistema genera attraverso il programma METAFONT quando ne ha bisogno o quando manca di quei font nel formato vettoriale (che sarebbe sempre il formato preferito e da preferire); qui vengono conservati anche i file metrici relativi a questi font generati con METAFONT. Poiché questa cartella ha la radice nella `$HOME` dell'utente, i file contenuti in questo albero sono specifici per un solo utente, il proprietario della cartella `$HOME`, ma non deve confondere questo albero con l'albero personale radicato in `$HOME/texmf`, di cui si è già parlato.

L'indicazione ⟨*radice*⟩ rappresenta il percorso che va dalla radice *del disco* su cui è installato il sistema TEX fino alla radice *dell'albero* che ci interessa; a seconda del sistema operativo potrà essere una cosa come `C:\`, oppure `/usr/share/`, oppure `/usr/local/`. Inoltre la doppia barra `//` dice al sistema TEX che quel percorso non si limita alle cartelle esplicitamente indicate, ma prosegue per tutte le ramificazione dell'albero fino alle ultime "foglie".

Vale la pena di ricordare la necessità di avere per ciascun albero il database dei nomi dei file sempre aggiornato; aggiornarlo quando non è necessario non produce nessun danno, ma omettere l'aggiornamento può costituire una fonte di frustrazione non indifferente perché il sistema potrebbe non essere in grado di trovare i file di cui necessita. Si ripete: *questo aggiornamento è particolarmente importante per l'albero personale con l'installazione MiKTEX.*

Tutto ciò può apparire estremamente complicato; in realtà l'utente o l'amministratore non hanno bisogno di occuparsene se non in circostanze eccezionali; è opportuno però conoscere l'esistenza di questa molteplicità di alberi e il loro contenuto. Una cosa importantissima da ritenere è che ad ogni installazione di un aggiornamento del sistema TEX vengono riscritti alcuni o tutti i file contenuti nelle cartelle la cui radice sia formata dalla sequenza di cartelle `texlive/`⟨*anno*⟩. Qualunque personalizzazione, quindi, non deve venire scritta dentro cartelle che si trovino dentro questi alberi, altrimenti se ne perde tutto il contenuto, perché verrebbe sovrascritta dai file di aggiornamento. Quando si installa una nuova versione, cambia l'⟨*anno*⟩ e quindi cambia tutta l'installazione; su alcuni sistemi è possibile far convivere diverse versioni del sistema TEX, ma questa pratica è consigliabile solo per utenti espertissimi che abbiano esigenze molto particolari.

4.4.2 Gli alberi di cartelle di MiKTEX

Sulle macchine Windows è possibile installare anche la distribuzione TEX Live del sistema TEX, ma spesso gli utenti delle macchine Windows installano la distribuzione MiKTEX.

I diversi sistemi operativi che vengono usati su queste piattaforme Windows sono molto variabili e non sempre completamente compatibili. Qui si farà riferimento ad una installazione con il sistema operativo Windows 7, che potrebbe essere ancora quello di prima installazione su diverse piccole macchine portatili; quanto si dice qui non dovrebbe essere sostanzialmente diverso da quello che potrebbe aversi su altri sistemi operativi più recenti come Windows 8 o Windows 10.

La moltitudine di alberi che si sono visti con la distribuzione TEX Live si ripete anche con MiKTEX, sebbene con un numero minore di alberi. Qui di seguito si indicherà con ⟨*radice*⟩ il percorso dalla radice *del disco* `X:\` *alla radice* dell'albero specifico che verrà descritto. I programmi eseguibili e della prima installazione non saranno necessariamente installati nel disco `C` e per questo qui si usa la lettera `X` per designare il disco su cui è installato il sistema TEX, anche se quasi sempre `X` coincide con `C`. In generale su Win7 si ha:

$$\langle \textit{radice} \rangle = \texttt{C:\Documents and Settings}$$

mentre su Win8 e sulle versioni successive del sistema operativo si ha:

$$\langle \textit{radice} \rangle = \texttt{C:\Users}$$

Si noti che nel seguito ⟨*versione*⟩ indica la versione della distribuzione MiKTEX, per esempio `2.9`.

1. `X:\Programmi\MiKTeX`⟨*versione*⟩`\` contiene tutti gli eseguibili del sistema TEX nella sottocartella `miktex\bin\`, ma contiene anche tutti i file che nella versione TEX Live sono contenuti negli alberi radicati in `textmf-dist` e `texmf-config`; in particolare i file di configurazione di default, i file di documentazione, e tutti gli altri file necessari per il funzionamento completo del sistema TEX.

2. ⟨*radice*⟩`\All Users\Dati Applicazioni\MiKTeX\`⟨*versione*⟩`\` contiene la struttura di cartelle contenenti i file generati dai programmi di configurazione e dai programmi di generazione dei font validi per tutti gli utenti della specifica macchina.

3. ⟨*radice*⟩`\`⟨*utente*⟩`\Impostazioni locali\Dati Applicazioni\MiKTeX` `\`⟨*versione*⟩`\` contiene i file di configurazione e quelli relativi ai font relativi all'uso dell'utente specifico.

4. ⟨*radice*⟩`\`⟨*utente*⟩`\Dati Applicazioni\MiKTeX\`⟨*versione*⟩`\` contiene i file di prima installazione e quant'altro sia necessario per il funzionamento del sistema TEX per l'utente specifico.

5. ⟨*radice*⟩\⟨*utente*⟩\`Documenti\` contiene l'albero personale di cui si è parlato precedentemente.

Non c'è una completa corrispondenza fra gli alberi di MiKTEX e quelli di TEX Live, ma non è difficile capire la strategia che sta sotto questa apparente complicazione; il documento `tds.pdf` spiega questa strategia illustrandone anche i pregi e i difetti, nonché i compromessi che è stato necessario raggiungere per avere un sistema funzionale e realizzabile.

Anche con MiKTEX l'utente non deve preoccuparsi più di tanto, se non nei rari casi in cui debba cercare un file specifico. Ricordi inoltre che il 'wizard' di MiKTEX per configurare il sistema, aggiornarlo, arricchirlo di altri pacchetti, eventualmente anche *on demand*[4], permette a MiKTEX di avere quelle funzionalità che lo rendono particolarmente gradito agli utenti dei sistemi operativi Windows.

4.4.3 L'aggiornamento dei database dei nomi dei file

L'argomento è talmente importante che si vogliono qui riassumere e ripetere le informazioni sparse nei paragrafi precedenti, anche per sottolineare questa questione cruciale.

Fintanto che si installa il sistema TEX, lo si configura, si installano nuovi pacchetti o se ne rimuovono alcuni di quelli già installati, si aggiornano sia gli eseguibili sia i pacchetti, ma sempre facendo uso del programma MiKTeX Settings o MiKTeX Maintainance per l'installazione MiKTEX per Windows, o TeX Live Manager per l'installazione TEX Live per tutti i sistemi, non è il caso di aggiornare il database dei nomi dei file, perché ci pensano quei programmi.

Va notato che aprendo MiKTeX Settings dal menù a tendina che viene attivato selezionando successivamente `Start|Programmi|MiKTeX`, si hanno due opzioni di 'manutenzione' (Maintainance): una da amministratore e una da utente. La prima serve per aggiornare i file di sistema e quelli a disposizione di tutti gli utenti della macchina, mentre il secondo serve per aggiornare solo i file a disposizione dell'utente specifico che sta usando il programma di manutenzione.

Il programma di manutenzione e di gestione dell'installazione TEX Live, tlmgr (che può operare sia in modo testuale dalla finestra comandi, sia in modo grafico interattivo), deve sempre venire usato con le prerogative dell'amministratore e,

[4] *On demand* non significa alla lettera 'a richiesta'. La cosa è molto più sottile: si immagini che l'utente stia compilando un documento che richieda l'uso di un certo pacchetto di estensione; l'utente lancia il programma di composizione, ma quando questo cerca di caricare quel pacchetto e non lo trova, controlla che faccia parte dei pacchetti disponibili per MiKTEX, ma non sia stato ancora installato; allora il programma di composizione si ferma, invoca il 'wizard' (il mago, lo stregone), che chiede all'utente se vuol installare il pacchetto mancante; alla riposta affermativa dell'utente il 'wizard' cerca in rete l'installatore di quel pacchetto, provvede all'installazione, all'aggiornamento dei file di configurazione, all'aggiornamento dei database dei nomi dei file e, quando ha terminato, restituisce il controllo al programma di composizione che prosegue usando anche il pacchetto appena installato. Non è una funzionalità da poco! Peccato che non funzioni se non si ha accesso alla rete; per questo e altri motivi si consiglia sempre di eseguire una installazione completa.

quando si attiva la funzione di aggiornamento dei database dei nomi dei file, li può aggiornare tutti cliccando un solo bottone.

Di solito non è necessario ricorrere a questo aggiornamento quando le modifiche ai file di sistema sono state eseguite attraverso i rispettivi 'manager'. Ma quando non è necessario aggiornare questi database oltre ai casi gestiti direttamente dal 'manager'? Solo in un solo caso, quello relativo all'albero personale quando si sia installata la distribuzione TEX Live, altrimenti *bisogna sempre aggiornare i propri alberi personali e gli alberi di sistema, se e quando si fossero installati dei pacchetti senza ricorrere ai 'manager'.*

Riepilogando, quindi, quanto detto sopra, bisogna sempre aggiornare i database dei nomi dei file nei casi seguenti:

1. Quando si aggiungono o si tolgono dei file dal proprio albero personale (installazione TEX Live *esclusa*).

2. Quando si aggiungono o si tolgono file dagli alberi che contengono informazioni per l'uso locale del sistema TEX.

3. Quando si installano negli alberi di sistema pacchetti di estensione o pacchetti di font senza usare i dovuti 'manager'.

L'ultimo caso non dovrebbe presentarsi mai, ma talvolta si ha a che fare con pacchetti sperimentali, non ancora previsti per l'installazione ufficiale; sarebbe preferibile inserirli nel proprio albero personale, ma talvolta (raramente) si rende necessaria l'installazione globale.

Ma negli altri casi *bisogna sempre aggiornare il database dei nomi dei file.* Non farlo metterebbe il sistema TEX nell'impossibilità di trovare i pacchetti che gli servono, e quindi verrebbero emessi messaggi d'errore la cui comprensione sembra essere difficoltosa, a quanto si deduce dalla lettura dei messaggi inviati al Forum del gruppo GUIT.

4.5 I programmi accessori

La composizione differita consente di usare un programma specializzato per ciascuna operazione di composizione, in modo da usare sempre il prodotto migliore per ciascuna fase di 'lavorazione' del documento.

I programmi accessori più importanti sono i seguenti:

1. Lo *shell editor* serve per generare e/o modificare i file sorgente da dare in pasto al programma di composizione. Se ne discuterà diffusamente ne prossimo paragrafo.

2. Benché il formato 'standard' DVI (DeVice Independent) del file di uscita del sistema TEX oggi sia usato molto raramente, esso era e rimane il formato di uscita più o meno predefinito; perciò è necessario disporre del programma specializzato per rendere il frutto della composizione visibile

e leggibile sullo schermo, o per renderlo stampabile con una varietà di dispositivi diversi, che possono andare dalle stampanti alle fotocompositrici. Ogni sistema TEX è corredato di un programma per visualizzare i file DVI, spesso utile anche per stampare. Talvolta è necessario convertire il file dal formato DVI al formato PostScript, e per gestire questo formato funziona egregiamente il programma dvips; ma per visualizzare il file così prodotto è utile disporre del programma ghostview, meglio conosciuto come gview o anche come gv; questo programma è una comoda interfaccia grafica per maneggiare un file PostScript al fine di visualizzarne il contenuto logico, per stamparlo e anche per trasformarlo in formato PDF. Chi esegue il lavoro vero è ghostscript, ma gview rende molto più agevoli tutte le operazioni, perché *ghostscript* è un poderoso programma, che però richiede di essere azionato dalla linea di comando (dalla finestra comandi di Windows o dalla finestra terminal, xterm, o console dei sistemi UNIX, Linux e MacOS X). Generalmente gli *shell editor* dispongono di un bottone sia per azionare il programma di visualizzazione per il formato DVI, sia per lanciare dvips, sia per visualizzare il file PS con il programma gview. Tutte queste operazioni oggi sono eseguite raramente, ma i programmi accessori per eseguirle fanno già parte di ogni distribuzione o sono annessi ad ogni distribuzione del sistema TEX.

3. I file in formato PDF, oggi preferito rispetto al formato DVI, sono maneggiabili mediante gli appositi programmi caratteristici di ogni sistema operativo; per tutti la Adobe produce il programma gratuito Adobe Reader, che consente di visualizzare i file nel formato PDF e di ottenerne alcune informazioni. Se si desidera la piena funzionalità sarebbe necessario acquistare da Adobe il programma Adobe Acrobat (eventualmente nella versione professional), oppure altri programmi simili; Adobe Acrobat consente di eseguire qualche ritocco al contenuto del file, di estrarne delle intere pagine o delle figure, di eseguire il ritaglio di parti della pagina, ovvero di scontornare le immagini; inoltre consente altre funzioni, in particolare la verifica della conformità di un file PDF alle norme relative all'archiviazione del documento elettronico; di questo si parlerà più diffusamente nel capitolo 22.

   Per le macchine Mac oltre a Adobe Reader (o Adobe Acrobat) il visualizzatore di sistema si chiama Preview (o Anteprima se il sistema operativo è localizzato in italiano); vale la pena di installare anche il visualizzatore Skim scaricandolo dal suo sito `http://skim-app.sourceforge.net/`; esso ha le prestazioni di Preview arricchite di molte altre utilissime funzioni, compresa quella di essere sincronizzabile con alcuni *shell editor*.

   Incidentalmente è bene puntualizzare: *sincronizzabile* è una espressione che può indurre a malintesi, ma è quella che viene usata per descrivere la funzione per la quale si può passare dalla finestra di editing del file sorgente al punto corrispondente della finestra di visualizzazione del file composto e viceversa. Questa funzione è fondamentale per facilitare la composizione di un documento; spesso essa è disponibile per i file composti con latex, che

produce il file di uscita in formato DVI; solo pochi editor e visualizzatori consentono la sincronizzazione per i file composti in formato PDF.

4. Sono utili anche diversi altri programmi di conversione di formato, non necessariamente già in dotazione del sistema operativo in uso, ma che sono comodi in diverse circostanze; per esempio `jpegtops` trasforma le immagini dal formato JPG (o JPEG) al formato PostScript, o meglio, Encapsulated PostScript. Il programma `ps2pdf` trasforma i file dal formato Encapsulated PostScript al formato PDF. Il programma `gimp`, nato per le piattaforme UNIX/Linux e ora disponibile anche per le altre piattaforme Windows e Mac, serve per modificare le immagini di quasi tutti i possibili formati e di salvarle in quasi tutti gli innumerevoli formati che è capace di gestire (però non può salvare in formati vettoriali – se ne parlerà diffusamente più avanti).

Questi programmi accessori sono funzionali di per sé e quindi possono essere usati anche indipendentemente dal sistema TEX, tuttavia risultano molto utili per svolgere alla perfezione certe funzioni che solo un programma specializzato sa fare, certamente meglio di quanto potrebbe fare un programma generico dalle 'troppe' funzioni.

Tutti i programmi menzionati, tranne Adobe Acrobat, sono disponibili gratuitamente e sono scaricabili dalla Rete adatti per essere usati sulle specifiche piattaforme di lavoro a disposizione. Adobe Acrobat ora funziona on line, senza scaricare niente, ma subordinatamente al pagamento di un abbonamento mensile.

4.6 Le tastiere

Questo è un argomento dolente per coloro che hanno la tastiera italiana. Nella tastiera italiana, figura 4.2, sono evidenziati i tasti normali e in particolare sono messe in evidenza le parentesi graffe che hanno una importanza enorme nella composizione dei file sorgente `.tex`.

Si nota però l'assenza dei segni ‘ (accento grave, o backtick, o virgoletta semplice aperta) e ~ (tilde, o legatura, o spazio insecabile). Per introdurre questi segni bisogna ricorrere a espedienti diversi a seconda della macchina usata.

Sulle macchine UNIX (Linux e Macintosh) non dovrebbero esserci grossi problemi, nel senso che esistono combinazioni di tasti che consentono di scrivere virtualmente qualunque carattere dell'alfabeto latino e vari segni e simboli usati normalmente negli scritti in italiano e nelle varie lingue occidentali.

Invece sulle macchine Windows esistono problemi seri che possono venire superati come spiegato qui di seguito.

4.6.1 Le tastiere delle macchine Windows

Il problema esiste con le macchine Windows, almeno fino a alla versione Win7 inclusa; la versione Win8 si comporta come le macchine Linux e Mac grazie ad

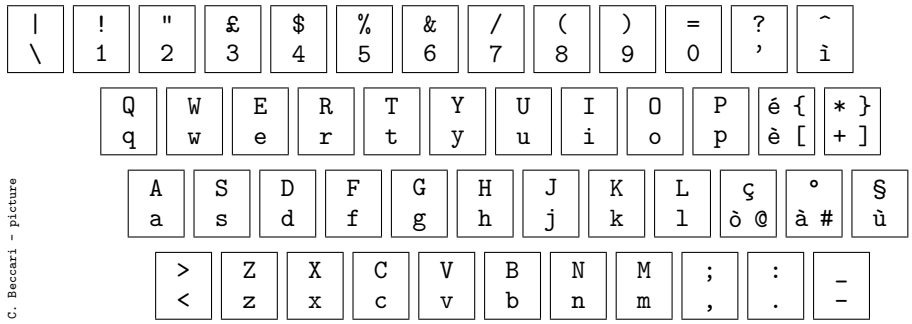


Figura 4.2: Schema della tastiera italiana sulle piattaforme Windows

opportune combinazioni di tasti premuti contemporaneamente o in sequenza. Con gli altri sistemi operativi si hanno alcune possibilità:

1. si tiene aperta la finestra dove si è lanciato l'applicativo Character map (o Mappa dei caratteri se il sistema operativo è in italiano) e ogni volta che occorre un segno non presente sulla tastiera lo si cerca in questa finestra, lo si seleziona e lo si copia, poi lo si incolla nel testo presente sullo schermo.

2. Si sfrutta lo *shell editor* per aprire dentro a questo la mappa dei caratteri disponibili con la codifica impostata per lo *shell editor*, si cerca il carattere assente dalla tastiera e lo si clicca; questo generalmente appare subito nel testo presente sullo schermo; ma può succedere, per esempio, che selezionando 'È' nella suddetta mappa dei caratteri disponibili, sullo schermo non appaia 'È', ma `\‘{E}`. Poco male; la lettura sullo schermo ci rimette un poco, ma l'operazione è corretta ed evita di dover andare a cercare il carattere ‘ assente dalla tastiera. Al massimo questa operazione può risultare un fardello per il correttore ortografico, ma ce ne sono che distinguono le sequenze TEX per gli accenti.

3. Disponendo di una tastiera con il tastierino numerico separato (o premendo il tasto `fn` che cambia significato ad un certo numero di tasti 'ordinari' in modo che si comportino come il tastierino laterale) basta premere il tasto `Alt` seguito da codice numerico ASCII[5] dell'accento grave, che per la cronaca è il numero 96 (oppure 096); rilasciando il tasto `Alt` dopo aver introdotto il codice numerico, sullo schermo appare il segno ‘. Si può usare lo stesso metodo per inserire la tilde il cui codice è 126. Per le maiuscole accentate e per gli altri segni si può procedere come nei punti precedenti oppure ci si scrive da parte una volta per tutte l'elenco dei codici delle lettere che si incontrano più sovente e si usa la tecnica qui descritta per

[5]I caratteri ASCII sono esposti nella tabella 26.1.

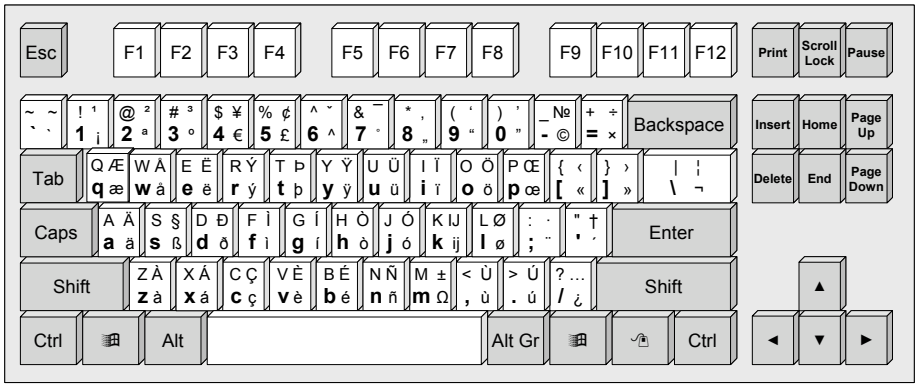


Figura 4.3: Schema della tastiera configurata con il driver EurKey. Tratto dal sito `http://eurkey.steffen.bruentjen.eu/`.

una introduzione abbastanza semplice che non necessita dell'uso del mouse, sgradito ad alcuni utenti.

4. Dal sito `http://www.microsoft.com/en-us/download/details.aspx?id=22339` si scarica il programma Microsoft Keyboard Layout Creator, che permette di configurare la tastiera come si preferisce, attivando tutte le combinazioni di tasti che si desiderano per introdurre tutti i segni disponibili con il driver della tastiera. Bisogna ricordarsi di dare un nome al layout personale che si è creato e di sceglierlo come tastiera predefinita. Molti utenti ridefiniscono il significato dei tasti che contengono segni raramente usati in italiano, come per esempio `ç` o `§` o `°` e sono contentissimi di questa personalizzazione. Le spiegazioni contenute in `http://www.gialloporpora.netsons.org/rimappare-la-tastiera-in-windows-creando-un-file-di-layout/395/` dovrebbero essere particolarmente utili.

   Vale la pena segnalare anche il sito `https://www.tastiera-aggiornata.it/` dove si descrive un layout di tastiera italiana estesa. Dal sito si può scaricare anche il software che permette di usare una tastiera italiana normale in modo esteso; l'idea è simile a quella esposta nella figura 4.3, ma i caratteri tradizionalmente assenti dalla tastiera italiana sono in posizioni diverse. Il software liberamente scaricabile dal sito è disponibile solo per piattaforme Windows con processore a 64 bit dalla versione Win7 in poi.

5. In un file di macro personali si inseriscono tutte le definizioni, le più mnemoniche possibile, per inserire i segni mancanti dalla tastiera; la definizione di nuovi comandi verrà vista molto più avanti, ma qui si indicano alcune definizioni il cui elenco può essere completato secondo le proprie necessità:

```
\newcommand\A{\‘A}
\newcommand\E{\‘E}
```

```
...
\newcommand\U{\‘U}
```

Come si vede si tratta di definizioni semplicissime, che possono venire ulteriormente arricchite di funzionalità, ma che consentono di evitare la complicazione di battere diversi tasti o di usare il mouse per inserire i segni assenti dalla tastiera italiana.

6. Per le macchine Windows e Linux esiste anche un driver EurKey scaricabile dal sito `http://eurkey.steffen.bruentjen.eu/`[6], che permette di avere un layout di tastiera molto più completo (sebbene anche il driver fornito insieme ai sistemi Linux sia già molto ricco). Il layout viene chiamato "layout europeo" perché dovrebbe essere in grado di introdurre qualunque segno latino con diacritici presente nelle lingue europee. Nella figura 4.3, tratta dal sito suddetto, vi sono mostrati i tasti con l'indicazione dei quattro segni che si possono comporre con ogni specifico tasto; si arriva a più di 200 segni che coprono quasi completamente le esigenze di tutte le lingue europee.

   Se si può segnalare un inconveniente, questo consiste nel fatto che il layout di base è quello USA, ma che i segni aggiuntivi non sono serigrafati né sulla tastiera USA, né su qualunque altra tastiera distribuita commercialmente. O si tiene il disegno della tastiera accanto al PC mentre si digita il codice di entrata, oppure ci si arrangia con piccole etichette da incollare sui tasti con un velo trasparente protettivo (scotch trasparente, per esempio), al fine di evitare la progressiva cancellazione delle indicazioni supplementari via via che si usa la tastiera.

   Si richiama il fatto che gli accenti morti servono anche per le lettere maiuscole; esiste persino un tasto, AltGr + m) che permette di passare alla digitazione in greco.

   L'uso di questa tastiera e dei suoi numerosissimi segni implicherebbe la specificazione dell'opzione *utf8* per il pacchetto *inputenc* nel preambolo di ogni documento LaTeX; ciò è superfluo dal 2018, quando la codifica UTF-8 è diventata quella preimpostata anche per pdfLaTeX.

Alternativamente, anche se la cosa può sembrare assurda, potrebbe essere conveniente pensare all'acquisto di una seconda tastiera; esistono diverse tastiere QWERTY nazionali o internazionali che consentono di introdurre tutti e 95 i caratteri ASCII premendo al massimo due tasti; se si è disposti ad abituarsi ad una disposizione dei tasti QWERTZ, le tastiere svizzere (figura 4.4) e tedesche sono complete; così come lo sono altre tastiere dei paesi del nord. Quelle inglesi e

[6] Questo tipo di driver esisterebbe anche per le piattaforme Mac; il sito di EurKey rimanda ad un repository `github` da cui è possibile scaricarlo, ma le istruzioni per l'installazione sono, forse, incomplete. D'altra parte i driver per le varie tastiere nazionali delle piattaforme Mac sono così completi che EurKey non sembra una estensione essenziale.

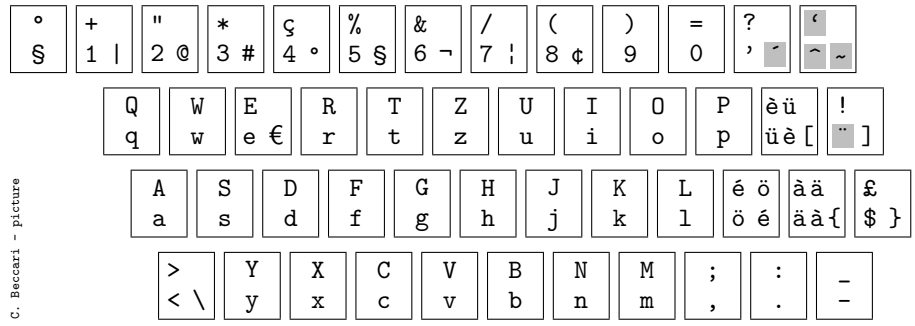


Figura 4.4: Schema della tastiera svizzera sulle piattaforme Windows. Vi si notano gli “accenti morti” evidenziati con lo sfondo grigio, i quali consentono di inserire il loro segno su qualunque lettera che lo accetti. La tastiera è utile per scrivere nelle quattro lingue ufficiali della Confederazione Elvetica, anche se non riporta tutte le vocali accentate dell’italiano e del romancio, che comunque si possono ottenere con gli accenti morti. Le maiuscole accentate si ottengono usando il tasto delle maiuscole permanenti.

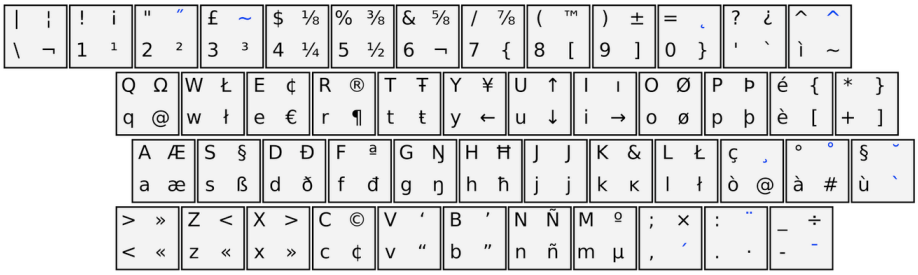


Figura 4.5: Schema della tastiera italiana su Linux senza personalizzazioni. La figura è tratta da Internet, ma il sito non è più raggiungibile.

irlandesi e quelle USA, specialmente se “professional” o “extended”, consentono di introdurre anche tutti o quasi tutti i segni previsti per la codifica ISO 8859-1. Spesso queste tastiere possono venire acquistate via Internet da produttori stranieri; spesso sono gli stessi produttori di computer a consentire l’acquisto via Internet e offrono le tastiere più disparate. Ovviamente è necessario che siano dotate del driver necessario per il sistema operativo della propria macchina, altrimenti sarebbero un acquisto inutile, anche se spesso si possono trovare in rete dei driver adatti ad una completa personalizzazione della tastiera.

Per una eccellente presentazione e descrizione delle tastiere nazionali e internazionali si veda il sito `http://en.wikipedia.org/wiki/Keyboard_layout`.

4.6.2 Tastiere Linux

Dopo l'installazione del sistema operativo Linux, in qualunque incarnazione, premere il tasto `AltGr` permette di ottenere diversi altri segni che sono normalmente assenti dalla tastiera italiana.

Dopo l'installazione del sistema la tastiera italiana potrebbe risultare arricchita di nuove funzionalità come appare nella disposizione indicata dalla figura 4.5. I segni marcati in blu sono i tasti morti per i segni diacritici; si noti tra l'altro la presenza sia della cediglia, sia della virgola sottoscritta del rumeno. Vi sono diverse ripetizioni, ma tutto sommato questa disposizione è piuttosto ricca di segni. In ogni caso nella impostazione di sistema che concerne la tastiera è possibile configurare le combinazioni di tasti che si preferiscono per agevolare l'introduzione dei segni che non sono direttamente marcati sui tasti.

L'uso di tutti questi segni implica la specificazione della codifica *utf8* al pacchetto *inputenc* nel preambolo di ogni documento LaTeX.

A questo proposito vale la pena di sottolineare che le distribuzioni del sistema TeX precedenti al 2018 producevano dei caratteri strani quando si ometteva di specificare la codifica usata per creare, modificare e salvare i file sorgente. Per XeLaTeX e LuaLaTeX, non è un problema, perché, come si ripeterà molte volte, i file sorgente da gestire con questi programmi devono essere creati, modificati e salvati con la codifica UTF-8, quindi per questi programmi UTF-8 è la sola codifica che si può usare[7].

Invece con pdfLaTeX, che non gestisce direttamente font UNICODE, ma solo polizze di caratteri codificati al massimo con 8 bit, prima del 2018 era necessario ricorrere al pacchetto *inputenc* per esplicitare la codifica usata. Con le distribuzioni del sistema TeX dal 2018 in poi, se i file sorgente sono salvati con la codifica UTF-8 non avvengono più pasticci strani, per esempio, con i caratteri accentati, non è più necessario specificare tramite *inputenc* la codifica usata, a meno che deliberatamente non si voglia usare una codifica diversa o non si debba ricomporre uno o più file creati prime dal 2018 con codifiche diverse. Non e un errore caricare *inputenc* con la specifica *utf8* anche se il file su cui si lavora è salvato con questa codifica, ma è decisamente superfluo.

4.6.3 Le tastiere sulle macchine Macintosh

Qui ci riferiamo alle macchine Macintosh moderne, dotate di sistema operativo Mac OS X. Queste macchine, qualunque disposizione di tasti abbiano (italiana, USA, o altra disposizione) dispongono di una comodissima interfaccia grafica che permette di avere sullo schermo la disposizione della tastiera e come questa cambia premendo i tasti serigrafati con i segni ⇧ (shift), ctrl, ⌥ (alt), o ⌘ (cmd). Nel disegno della tastiera i tasti morti sono disegnati in colore beige; dopo aver premuto (o cliccato su) un tasto morto la tastiera normale cambia

[7] In realtà non è vero; si possono usare anche caratteri Type 1 che non sono conformi con la codifica UTF-8, ma bisogna procedere con molta cautela; se ne riparlerà a tempo debito più avanti.

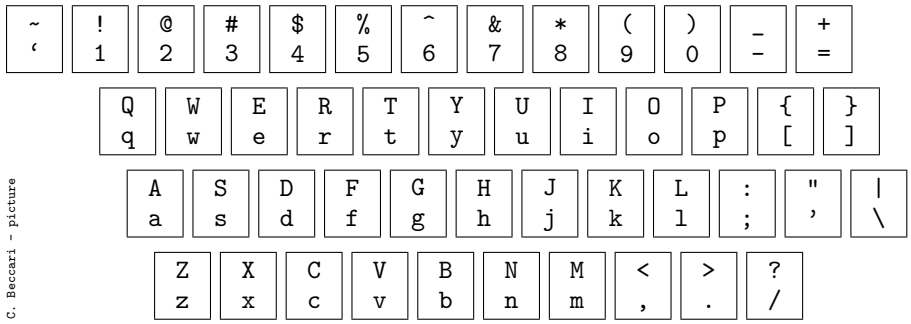


Figura 4.6: Schema della tastiera USA estesa sul MacBook Pro

aspetto e presenta solo i caratteri che corrispondono alle lettere ‘accentate’ con il particolare ‘accento’ del tasto morto usato. Dopo un poco si imparano a memoria queste combinazioni di tasti e non è più necessario ricorrere all’immagine della tastiera.

Chi scrive usa abitualmente una tastiera estesa con la disposizione dei tasti secondo lo schema USA, priva, quindi, di qualunque segno accentato, come si vede nella figura 4.6; sono disponibili due driver, quello ‘USA’ e quello ‘USA extended’, il secondo driver essendo il driver più performante. Tuttavia chi scrive ormai esegue automaticamente l’operazione, per esempio, di battere assieme i tasti alt e ‘ e successivamente i tasti shift e E per avere nello schermo il carattere ‘È’.

4.6.4 Le tastiere virtuali

Abbiamo visto sopra che sulle macchine Mac esiste un applicativo che permette di mostrare il layout della tastiera in uso e che permette di scrivere anche i segni che non sono serigrafati sui tasti. Lavorando con la mano sinistra sui tasti di controllo e con la mano destra sul mouse si riesce a scrivere praticamente con ogni tastiera il cui driver sia stato installato nel sistema. Questo è particolarmente utile per scrivere anche in alfabeti diversi da quello al quale corrisponde la tastiera fisica connessa all’elaboratore. Ecco, quindi, come si può agevolmente scrivere in cirillico o in greco. Ma apparentemente questo si può fare solo sulle macchine Macintosh.

Non è così: chi scrive non dispone di una macchina Windows, ma è al corrente che con i sistemi operativi 8 e seguenti anche su queste macchine esista la possibilità di visualizzazione a schermo della tastiera interattiva come sui Mac. Inoltre oggi si stanno diffondendo gli schermi *touch screen*.

Per lo più questa funzionalità *touch screen* viene riservata ad alcuni giochi o a operazioni grafiche. Ma questi elaboratori possono davvero funzionare senza una tastiera materiale, consentendo all’utente di servirsi del disegno della tastiera

riportato sullo schermo; l'utente tocca i tasti virtuali sullo schermo e il testo appare nel documento che si sta scrivendo. Estremamente comodo e funzionale, anche se costringe l'utente a scrivere in una posizione innaturale, su una tastiera sostanzialmente verticale, come lo schermo, e non orizzontale come una tastiera materiale.

Tuttavia se si dispone di una tastiera italiana ma si desidera usare una tastiera USA, per esempio, basta averne caricato il driver, e averlo azionato toccando l'apposta icona nella barra inferiore o superiore dello schermo. Se si vuole usare una tastiera greca, lo stesso, basta averne caricato il driver e aver toccato la sua icona per attivare quella tastiera. E così vale per ogni tastiera con uno schema dei tasti diverso da quello della tastiera fisica in uso.

Oggi poi si stanno diffondendo dei leggeri e non tanto piccoli elaboratori portatili dotati di schermo *touch screen*; essi possono anche essere privi di una tastiera fisica. Se dispongono di una tastiera (*tablet* PC), si può ruotare lo schermo in posizione orizzontale, coricandolo sopra la sua tastiera, e usarlo per scrivere con le mani nella posizione solita ed ergonomicamente valida. Questi piccoli elaboratori possono essere usati tranquillamente non solo come tastiere virtuali, ma se sono di potenza adeguata (disco, RAM, velocità, dimensioni dello schermo) possono anche sostituire completamente un PC "normale", sia esso da tavolo o da viaggio.

Il problema tastiera con questi dispositivi cessa di essere un problema.

4.7 Gli shell editor

Nelle versioni precedenti alla 0.90 di questa Introduzione non si era deliberatamente fatto nessun cenno agli *shell editor* perché si riteneva che questo fosse un argomento facoltativo, non direttamente collegato alla composizione tipografica con LaTeX.

In un certo senso questo argomento è ancora valido; in rete e in commercio esistono tanti diversi *shell editor* adatti ai vari sistemi operativi che forse c'è addirittura l'imbarazzo della scelta; tuttavia si è ora ritenuto che fosse necessario dedicare un cenno a qualche *shell editor*, specialmente quelli che sono direttamente disponibili con i software di installazione del sistema TeX.

Di alcuni *shell editor* basta conoscere il nome, cercare su Internet, e scaricare dall'opportuno sito il programma di installazione, provvedere all'installazione e poi farne uso in modo da imparare progressivamente a sfruttarli al meglio.

Essi hanno in comune alcune caratteristiche:

- permettono di creare e correggere (editare, in gergo) file puramente testuali, non formattati, o comunque di salvare il file in modo che contenga solo il testo e il mark-up, senza nessuna informazione sulla formattazione del testo, perché questa operazione è rimandata alla compilazione sotto l'azione dei file di classe e dei file di estensione usati. Quasi tutti permettono di scegliere fra diverse codifiche dei caratteri immessi da tastiera; i più recenti

e meno caratteristici di una particolare piattaforma consentono l'uso della codifica UNICODE. Si parlerà diffusamente più avanti in diversi capitoli della necessità di scegliere opportunamente la codifica per i file sorgente (vedi il capitolo 26).

- Durante il processo di editing è possibile avere in azione uno o più correttori ortografici per una o più lingue. Inoltre alcuni editor possono eseguire operazioni di taglia e incolla non solo per gruppi di righe, ma anche per blocchi rettangolari di un certo numero di righe che si estendono orizzontalmente per un certo numero di colonne; con questi editor il mouse ha anche delle funzioni di selezione più estese di quelle che sono normalmente disponibili.

- Per quel che riguarda il testo da comporre è generalmente possibile scegliere la codifica (encoding) dei caratteri introdotti con la tastiera e, se questa è di tipo avanzato, è anche possibile scegliere la codifica UNICODE, al punto da poter introdurre direttamente caratteri di altri alfabeti, anche retrogradi, e gli ideogrammi delle lingue dell'estremo oriente.

- Se si sceglie di suddividere il file del documento in tanti spezzoni di dimensioni minori, essi generalmente consentono di editare diversi file alla volta.

- Essi generalmente consentono di scrivere e salvare su disco i file sorgente per qualunque programma del sistema TEX; come si vedrà, i programmi di composizione di questo sistema non si limitano al solo LATEX, e non si limitano solamente alla composizione di testi, ma anche alla creazione di disegni vettoriali o a matrici di punti, quali sono, ad esempio, gli stessi caratteri con cui vengono composti i testi.

- Permettono di cliccare su un bottone virtuale con il puntatore del mouse, bottone scelto fra una serie di bottoni per scopi analoghi, oppure di scegliere il programma di composizione da una lista di un menù a caduta, per salvare il file e per lanciare direttamente il programma di composizione.

- Dopo l'esecuzione del programma di composizione, se non ci sono stati errori, lo *shell editor* apre direttamente il programma di visualizzazione adatto al formato di uscita del documento composto. Se invece ci sono errori, lo *shell editor* permette di aprire il file "di registro" nel quale sono elencati gli errori in modo da mettere in relazione le righe diagnosticate come erronee, come appaiono nel file di registro, con le righe presenti nel file sorgente così da poter navigare facilmente avanti e indietro fra il file di registro e il file sorgente.

- Nello stesso tempo se non ci sono errori di mark-up, ma il testo composto presenta dei refusi o delle frasi incomplete, erronee, imprecise, oscure, o altro, lo *shell editor* permette di navigare avanti ed indietro fra la finestra di visualizzazione del documento composto e la finestra dell'editor contenente

il file sorgente, in modo da usare la finestra di visualizzazione come se fossero le bozze del documento così da apportare le debite correzioni o modifiche al file sorgente. In genere questa operazione su alcune piattaforme risulta più difficile o impossibile con il formato PDF, ma è sempre possibile con il formato DVI, anche se poi il documento finale verrà prodotto con il formato PDF. Sui sistemi Mac questo inconveniente non si presenta mai perché il programma di visualizzazione integrato in TeXShop è fatto apposta per gestire agilmente i file in formato PDF, che rappresentano il formato di default di tutti i documenti presenti sulle macchine Mac. Ora però sono disponibili alcuni editor multipiattaforma che consentono di spostarsi agevolmente fra punti corrispondenti della finestra di composizione contenente il file sorgente e della finestra di visualizzazione anche con il formato PDF.

- Alcuni *shell editor* sono multipiattaforma, vuoi perché sono scritti in un linguaggio interpretato, per esempio Java, vuoi perché vengono prodotti simultaneamente nelle versioni adatte alle tre piattaforme principali.

- Alcuni *shell editor* sono freeware o, comunque, gratuiti, altri sono shareware; le distribuzioni commerciali del sistema TEX contengono *shell editor* specifici della particolare versione commerciale e quindi sono, per così dire, compresi nel prezzo.

- Alcuni programmi non sono dei veri e propri *shell editor*, ma consentono di visualizzare direttamente il prodotto della composizione oppure permettono di presentare schermate che assomigliano fortemente al risultato finale, nel senso che i comandi di mark-up (almeno alcuni) sono già interpretati in modo da mostrare sulla finestra dell'editor praticamente solo il testo. Di questi programmi si farà nel seguito un breve cenno separatamente dalle brevi descrizioni relative agli *shell editor* veri e propri.

Il sito `http://en.wikipedia.org/wiki/Comparison_of_TeX_editors` è di grande interesse. Vi si trova una grande tabella dove sono nominati virtualmente tutti gli *shell editor* esistenti. Di ciascuno viene detto se si tratta di software gratuito o se è un programma commerciale; per quali piattaforme esiste il programma; se assomiglia ad un editor WYSIWYM (vedi poco più avanti) o se lavora solo sul file sorgente; in più, ovviamente, vengono elencate tutte le caratteristiche illustrate sopra insieme ad altre caratteristiche meno essenziali. Il confronto è molto istruttivo e permette di scegliere il proprio *shell editor* a ragion veduta. In generale quelli che accompagnano nativamente ogni distribuzione sono abbastanza buoni, talvolta ottimi. Apparentemente, però, quel sito lascerebbe intendere che esista un solo editor multipiattaforma, che, se dotato dell'opportuno plug-in, permette di avere a portata di mano tutte le possibili caratteristiche che si possono chiedere ad uno *shell editor*. Chi scrive consiglia di non prendere quelle informazioni per oro colato: si tratta di informazioni utilissime, ma talvolta incomplete, quindi il lettore si serva pure di quella tabella, ma con giudizio.

4.7.1 Shell editor multipiattaforma

Come già accennato esistono *shell editor* multipiattaforma: lo stesso programma gira su macchine diverse, con hardware e software diversi. Questo è possibile se i programmi relativi sono scritti in un linguaggio interpretato di cui esista l'interprete per tutte le piattaforme, per esempio, il linguaggio Java con la sua Java Virtual Machine.

Altri *shell editor* sono multipiattaforma nell'altro senso, cioè chi li ha prodotti ha compilato i codici sorgente in modo da ottenere i codici eseguibili validi per tutte e tre le piattaforme principali; a parte questo dettaglio tecnico, essi hanno anche lo stesso aspetto da un punto di vista grafico ed hanno esattamente le stesse prestazioni, gli stessi comandi, le stesse scorciatoie, eccetera.

Alcuni sono multipiattaforma nel senso che chi li ha prodotti ha provveduto a generare i file di installazione che ne permettono l'uso immediato, ma solo per due sistemi operativi, mentre per il terzo sono necessarie procedure particolari per l'installazione; in ogni caso l'aspetto grafico e il funzionamento sono sostanzialmente identici.

emacs è un potentissimo *shell editor* molto noto agli utenti dei sistemi UNIX; associato alla sua estensione `auctex` permette di editare file LaTeX con una serie di facilitazioni ben congegnate; questa estensione consente anche di eseguire la compilazione e la successiva previsione dell'aspetto grafico di parti selezionate del file sorgente. In pratica, selezionando una parte del file sorgente e cliccando su un opportuno bottone (o dando da tastiera uno dei numerosissimi comandi abbreviati ottenuti premendo contemporaneamente uno o più tasti di controllo (Ctrl, Shft, Alt, ...) e un altro tasto (lettera, cifra, ecc.)), si può compilare e vedere nel visore del file di uscita quel brano già composto come lo farebbe latex; questa funzionalità è molto utile specialmente quando si devono comporre formule complicate, tabelle complicate, disegni, eccetera.

emacs consente di usare tutte le possibili codifiche dei font in entrata, anche se con le macchine Windows vecchiotte converrebbe impostare l'encoding *ansinew*.

Questo editor è potentissimo, ma molti utenti lo trovano difficile da usare; esso diventa formidabile quando si conoscono a memoria moltissimi comandi abbreviati, cosicché non sia quasi mai necessario ricorrere al mouse, che i puristi non usano mai.

TeXstudio ha ottime caratteristiche per la gestione dei file sorgente; permette di vedere la struttura del documento, di disporre di menù a discesa per selezionare i vari simboli matematici, i vari ambienti, i caratteri non ASCII, con comodi pulsanti a lato della barra a sinistra del panello dell'editor. Consente di fare il passaggio dalla finestra del file sorgente alla finestra del file composto e viceversa, purché il commando per lanciare pdflatex contenga l'opzione `-synctex=1`.

La prima volta che si usa TeXstudio, esso fa una scansione del sistema per vedere quali strumenti siano già installati e li inserisce direttamente nella configurazione del programma, configurazione che in un secondo tempo l'utente può cambiare a piacimento, in particolare per quel che riguarda l'encoding dei caratteri in entrata.

Chi scrive l'ha usato e trova che sia molto buono. Esso può essere fatto lavorare anche con il visualizzatore esterno SumatraPDF con il quale è possibile sincronizzare le finestre di composizione e di visualizzazione come con il visualizzatore interno, ma lasciando inalterate tutte le altre funzionalità di SumatraPDF. Si tenga presente che il formato PDF diventerà de facto lo standard per la composizione dei documenti.

Per quel che riguarda il riconoscimento della codifica esso è automatico per la codifica *utf8*, ma quando il file da aprire non è scritto con questa codifica esso se ne accorge e apre una finestra di dialogo perché l'utente scelga la codifica opportuna. Tutto ciò non è gestibile con le righe di autoconfigurazione; si veda la descrizione di TeXShop nella pagina 117.

Texmaker è il programma originale da cui è nato TeXstudio; hanno in comune un certo look generale delle finestre e molte funzionalità; quello che (per ora) manca a Texmaker è la capacità di interpretare i commenti di autoconfigurazione; sono talmente utili che questa mancanza può essere accettata solo da chi non ne conosce l'utilità; si veda la descrizione di TeXShop nella pagina 117.

TeXworks è un programma freeware multipiattaforma derivato dal programma TeXShop per i calcolatori Mac; è uno *shell editor* la cui versione stabile viene installata direttamente sulle macchine Windows e Mac quando si installa TEX Live o MacTEX rispettivamente; per le macchine Linux bisogna installarlo appositamente come spiegato qui appresso. In ogni caso si possono scaricare dal suo sito le versioni in evoluzione per Windows e Mac; per installare TeXworks sulle macchine Linux bisogna scaricare il pacchetto dei file sorgente e bisogna completare le librerie in dotazione al sistema; poi diventa abbastanza facile seguire le indicazioni nel sito `http://code.google.com/p/texworks/wiki/Building` per la particolare versione del sistema operativo Linux che si sta usando.

Per gli utenti della distribuzione MiKTEX non c'è per ora nessun suggerimento che il programma sia già disponibile; chi ne ha sentito parlare, lo trova già lì, nella cartella `.../MiKTeX2.9/miktex/bin/`; se si vuole, se ne può creare una "scorciatoia" da trasferire sul desktop, in modo da poterne fare l'uso che si desidera. La prima volta che lo si usa esso cerca i file di cui ha bisogno e configura tutto o quasi quello che serve.

TeXworks è in grado di usare l'encoding UNICODE e di lanciare tutti i principali programmi di composizione del sistema TEX. Il suo visualizzatore integrato, però, visualizza direttamente i file di uscita in formato PDF e

permette di passare direttamente con un semplice click del mouse dalla finestra di composizione al punto corrispondente della finestra di visualizzazione e viceversa; il visualizzatore è molto rapido. Le dimensioni delle due finestre sono adeguate a riempire bene uno schermo con rapporto di forma 16 : 9, o altri simili schermi panoramici, in modo da averle affiancate senza sovrapposizioni.

Dispone di un comodo correttore ortografico usabile per diverse lingue; solo il dizionario ortografico per l'inglese è preinstallato; per l'italiano bisogna scaricare lo stesso file di correzione ortografica che usa OpenOffice.org e metterlo in una apposita cartella, e così bisogna fare per le altre lingue il cui dizionario ortografico non sia già preinstallato.

Con TeXworks, installato con la distribuzione di MiKTEX 2.9, i correttori ortografici preinstallati per altre lingue potrebbero essere aumentati: oltre all'inglese americano potrebbero esserci anche i file di correzione ortografica per il francese e per il tedesco; se si vuole aggiungere la possibilità di correggere anche l'italiano, o qualunque altra lingua, bisogna aggiungere i file `it_IT.aff` e `it_IT.dic` in una cartella particolare, che se non è ancora presente bisogna creare; sulle macchine Windows recenti dovrebbe essere presente una cartella `C:\Users\`⟨*utente*⟩`\TeXworks` che contiene altre cartelle; in questa cartella `TeXworks\` bisogna creare la nuova cartella `dictionaries\` e vi si devono salvare i due file summenzionati; se si vogliono altre lingue, si aggiungono i corrispondenti file `.dic` e `.aff` con le sigle delle lingue simili a quelle dell'italiano: `fr_FR` per il francese, `en_US` per l'inglese americano, eccetera. Quando si apre TeXworks esso riconosce la presenza dei file in questione, così che diventano selezionabili per la verifica ortografica del testo italiano o delle altre lingue.

TeXworks consente di interpretare direttamente certe speciali linee di commento all'inizio dei file sorgente, in modo da poter specificare sia il programma di composizione desiderato, sia la codifica del particolare file, sia, importantissimo, il nome del file principale, il master file, in modo che il progetto possa essere gestito senza file ausiliari; si veda la descrizione di TeXShop nella pagina 117; sebbene TeXworks accetti sintassi di autoconfigurazione leggermente diverse, esso interpreta correttamente anche quelle di TeXShop. Questo *shell editor* dispone anche della prerogativa del completamento dei comandi: si comincia a scrivere un comando, poi si batte il tasto Tab o ⟶ finché sullo schermo non appare il comando completo desiderato. Nel file di Aiuto sono elencate anche le numerose scorciatoie che esso consente di usare quando si desidera evitare l'uso del mouse. Certamente in poco tempo verranno messi in luce sia gli errori, sia le prestazioni ancora mancanti e, per quando sarà arrivato alla versione 1.0 (probabilmente anche prima), potrebbe diventare de facto l'unico *shell editor* per tutte le piattaforme.

Questi *shell editor* multipiattaforma sono i più consigliabili in generale; se

poi uno desidera avvalersi di prestazioni particolari su una macchina particolare, allora può considerare la possibilità di installare uno *shell editor* specifico fra quelli descritti nelle sezioni seguenti.

4.7.2 Shell editor per le macchine Windows

Per le macchine Windows il disco di installazione TEX Live offre la possibilità di installare il freeware TeXnicCenter; il programma di installazione del sistema MiKTEX, ProTEXt, segnala la possibilità di installare lo shareware WinEdt o il freeware Texmaker indicandone i siti da cui eseguire il download. In rete si trova anche gli editor freeware TeXstudio e LED. Esiste anche una versione del freeware emacs per le macchine Windows.

TeXstudio È uno *shell editor* multipiattaforma ed è già stato descritto nel paragrafo 4.7.1. Va notato che ultimamente viene installato insieme a MiKTEX che, invece, precedentemente installava TeXnicCenter.

Texmaker È uno *shell editor* multipiattaforma ed è stato già descritto nel paragrafo 4.7.1; alcuni lo preferiscono a TeXstudio, forse perché non hanno ancora scoperto l'utilità dei commenti di autoconfigurazione, che invece TeXstudio riconosce.

TeXworks È uno *shell editor* multipiattaforma ed è stato già descritto nel paragrafo 4.7.1; come si è già detto questo programma riconosce le righe di autoconfigurazione come TeXstudio. Esso è più spartano di TeXstudio ma molti lo preferiscono per la pulizia della schermata non ingombrata da numerose barre di icone.

WinEdt è un programma shareware ottimo, specialmente sviluppato per le macchine Windows del cui sistema operativo sfrutta non poche librerie dinamiche; usa anche la codifica dei font in entrata specifica delle macchine Windows su cui opera, offre al "tastierista" la possibilità di immettere una grande quantità di caratteri non rappresentati direttamente sulla tastiera, segni speciali e lettere con ogni possibile segno diacritico; permette di introdurre quasi tutti i segni compresi nello standard ISO 8859-1 (Latin 1); le versioni più recenti possono anche usare la codifica UNICODE; quindi nel preambolo dei documenti da comporre è necessario specificare l'opzione della codifica da usare, sia essa *latin1* o *utf8*.

Il programma permette di disporre di una moltitudine di menù che consentono con un solo click del mouse di introdurre nel testo sorgente le macro che individuano i simboli e i segni degli operatori matematici di vario genere.

Esso permette di disporre di maschere (*template*) per ogni genere di ambiente testuale, grafico o matematico, cosicché inserita la maschera nel file sorgente non resta che riempirla con le informazioni specifiche del materiale da comporre.

Così come è distribuito, il programma dispone di una moltitudine di vocabolari per la verifica ortografica; per motivi storici la lista di parole che compongono il dizionario per la lingua italiana è molto ricco, ma le parole tronche sono prive di accento; questo apparente inconveniente viene eliminato con l'uso, perché ogni volta che una parola tronca correttamente accentata (e, ovviamente, anche ortograficamente corretta) presente nel file sorgente viene segnalata come errata, basta un click di mouse per aggiungerla al dizionario; dopo pochi giorni d'uso le parole tronche che l'autore usa abitualmente sono tutte inserite nel dizionario; successivamente la mancanza di altre parole tronche viene segnalata solo raramente.

Il programma permette di definire un "progetto", vale a dire di definire in una struttura interna, in un file con estensione `.prj`, l'elenco completo dei file che compongono un dato documento e permettono di registrare anche le impostazioni per il o i dizionari da usare per quel documento, così come altre impostazioni in merito, per esempio, alla gestione delle righe che compaiono troppo lunghe per restare completamente nello schermo.

TeXnicCenter offre prestazioni simili ma meno estese di quelle che si possono avere con WinEdt; gli utenti lo trovano (oltre che gratuito) più semplice da usare di WinEdt, ma le prestazioni sono, secondo chi scrive, decisamente inferiori. Recentemente sembra che la manutenzione di questo programma sia cessata, tanto che prima veniva installato di default insieme a MiKTEX, mentre ora le installazioni di MiKTEX includono TeXstudio, oltre a TeXworks.

LED è un programma freeware con alcune caratteristiche particolari, come per esempio quella di sfruttare bene gli schermi "panoramici" dei moderni PC e laptop; infatti è possibile tenere aperte simultaneamente una accanto all'altra le finestre del testo sorgente e quella del testo composto (solo però in formato DVI), cosicché la navigazione da una finestra all'altra avviene in modo semplicissimo e comodissimo per controllare errori, refusi, imprecisioni o altro. La schermata del testo composto viene aggiornata automaticamente ogni volta che si clicca sul bottone per l'esecuzione della composizione in formato DVI, che, data la velocità dei PC moderni, dà quasi l'illusione di una composizione sincrona.

Inoltre le funzionalità di LED sono simili a quelle degli altri programmi già descritti; purtroppo non dispone di un visore diretto di file in formato PDF, per cui bisogna ricorrere ad un programma esterno, per esempio Adobe Reader, che però non si integra perfettamente con il programma LED, non altrettanto bene quanto il suo visore interno per i file in formato DVI, cosicché si perdono un poco le prestazioni che LED offre con quest'ultimo formato.

TextPad è uno shareware leggermente più economico di WinEdt (32 € contro 40 €) e anche leggermente meno performante; molti sviluppatori di software lo usano perché permette di gestire file anche di grandi dimensioni

(contrariamente al piccolo Notepad di serie su ogni macchina Windows) ma TextPad garantisce una piccola "invasività" della memoria RAM. Ne esiste anche una versione localizzata in italiano, cosicché, almeno per lo *shell editor*, gli utenti poco ferrati in inglese si sentono a loro agio.

Kile è uno *shell editor* che opera solo su macchine dispongano dell'interfaccia grafica KDE (vedi poco più avanti); per le macchine Windows esiste il programma CygWin che permette di emulare il sistema operativo UNIX/ Linux all'interno di un apposito ambiente grafico; dentro questo ambiente CygWin è possibile installare una versione dell'interfaccia grafica KDE, per cui anche sulle macchine Windows si può usare Kile. Oggi poi esiste un installer di Kile per Windows che provvede a creare una specifica interfaccia KDE, quindi l'installazione diventa molto più semplice.

4.7.3 Shell editor per le macchine Linux

Con le macchine Linux si ha una grande disponibilità di programmi adatti a scrivere ed editare file testuali, anche perché in questo sistema si fa un grande uso di *scripting*, cioè di programmi scritti senza particolari formattazioni, anzi, spesso contenenti solo i caratteri da 32 a 127 della pagina ASCII dei caratteri[8].

Gli utenti fortemente coinvolti con una macchina Linux e che usano lo scripting in modo esteso, generalmente si servono del programma vim oppure del programma emacs. Per usare un vero *shell editor* appositamente previsto per l'uso con LaTeX, si può ricorrere al programma Kile, purché l'interfaccia grafica sia l'interfaccia KDE; attenzione, però; su una vera piattaforma Linux, Kile necessita della presenza di un installazione TeX Live conforme alle specifiche del consorzio Debian.

vim è un programma con una interfaccia grafica virtualmente assente; molto veloce e molto efficace nelle mani di un utente di Linux molto esperto, ma poco utile ad un utente "normale". È solamente un efficacissimo "text editor", ma non lo si può chiamare "shell editor" nel senso stretto della parola. Esiste una versione con interfaccia grafica che è più facile da usare, ma resta sempre solo un editor di testi.

emacs è già stato descritto come programma multipiattaforma e non sarebbe necessario aggiungere altro. Merita invece segnalare che emacs è installato di default in ogni distribuzione delle varie implementazioni del sistema operativo Linux; quindi ogni utente Linux ne dispone per il solo fatto di usare una macchina con sistema operativo Linux.

[8] Sarebbe la prima mezza pagina di ogni codifica di caratteri che contenga solo 256 caratteri, oppure la prima (mezza) pagina delle ulteriori numerose pagine ognuna di 256 caratteri, come la codifica UNICODE. La codifica ANSI contiene una unica pagina di 256 caratteri; le varie codifiche ISO 8859 contengono ciascuna una sola pagina di 256 caratteri. Per tutte le codifiche, però, la prima mezza pagina della prima o dell'unica pagina contiene i caratteri ASCII.

Conviene ricordare di usare sempre il plug-in `auctex` che fa diventare emacs un vero *shell editor* per LaTeX, oltre a consentire la visualizzazione quasi sincrona di tratti selezionati del file sorgente.

Xemacs è una versione di emacs più avanzata e che già contiene alcune delle prerogative del plug-in auctex, il quale è ugualmente adatto anche a Xemacs. emacs è normalmente installato insieme al sistema Linux; Xemacs invece va installato deliberatamente. Esiste una versione di Xemacs anche per le macchine Windows.

Kile per i sistemi Linux che dispongono del "K Desktop Environment", in sigla KDE, è disponibile anche lo *shell editor* Kile: KDE Integrated LaTeX Environment; è dipendente dalla presenza di una installazione TeX Live, certamente quella Debian compatibile sulle macchine con una versione Debian di Linux; non è escluso che qualsiasi installazione TeX Live possa essere sufficiente se si è fatto "credere" al sistema operativo che sia già installata la versione Debian di TeX Live. Si tratta di un ottimo *shell editor* con tutte le funzionalità necessarie e già descritte per altri *shell editor*; in più esso ha indicazioni grafiche che permettono di controllare gli ambienti testuali o matematici, in modo che si possa essere sicuri che ogni apertura di ambiente abbia la corrispondente chiusura (questo è il minimo che si può richiedere e, in un modo o nell'altro, quasi tutti gli *shell editor* offrono questa funzionalità) ma permette anche di contrarre il contenuto di ciascun ambiente, cioè si può ridurre sullo schermo la visualizzazione di tutto l'ambiente ad una sola riga, in modo che lavorando sul file sorgente si possa esaminare con più semplicità il testo che precede e quello che segue l'ambiente; cliccando su una piccola icona a forma di ⊞, l'ambiente viene nuovamente espanso; si tratta di una funzionalità grafica molto utile.

Kile permette anche di sfruttare il completamento dei nomi; vale a dire che quando si comincia a scrivere una macroistruzione, per esempio, il programma dopo qualche carattere presenta un menù a discesa con tutti i nomi che cominciano con quelle lettere e fra i quali si può scegliere il nome giusto. Per gli ambienti non appena si scrive lo statement di apertura, Kile inserisce subito dopo lo statement di chiusura; se l'ambiente richiede obbligatoriamente dei parametri, esso inserisce anche le parentesi graffe che li racchiuderanno; sta poi al "tastierista" di completare il gruppo dei parametri e di inserire il contenuto dell'ambiente. Ad alcuni questa funzionalità piace moltissimo, e in generale è considerata una funzionalità molto utile per qualsiasi editor; altri ne sono un poco infastiditi; ovviamente si tratta di gusti personali, per venire incontro ai quali Kile permette di configurare la finestra di editing con o senza questa funzionalità.

Quando si esegue una composizione, diciamo con pdflatex, Kile provvede direttamente a ripetere la compilazione quante volte occorre affinché tutti i riferimenti incrociati siano corretti, che la bibliografia contenga tutti i riferimenti correttamente collegati con i richiami, che l'indice analitico

sia aggiornato, eccetera. Anche questa funzionalità può venire disabilitata per la lavorazione iniziale di un documento, e riattivata nella fase finale, cosicché si possa essere sicuri che il documento sia globalmente completo e corretto.

Le macchine Linux che operano con l'interfaccia grafica Gnome possono comunque attivare una parte ridotta dell'interfaccia KDE in modo da poter usare ugualmente Kile, oppure possono ricorrere al programma seguente.

gedit-LaTeX-plugin gedit è il semplice editor testuale con interfaccia grafica disponibile sulle macchine Linux con il desktop gnome. Questo semplice editor accetta l'innesto di plug-in, e il gedit-LaTeX-plugin è quello che permette di gestire i file LaTeX attraverso la sua semplice interfaccia grafica. Ma questo plug-in offre numerose prestazioni utili per la gestione, la modifica e l'esecuzione/compilazione dei file LaTeX. Precisamente consente di completare i comandi; controlla la sintassi del mark-up di LaTeX; permette di avere come Kile, Texmaker e TeXstudio una finestra laterale con la struttura di un documento complesso e permette di navigare facilmente da un punto all'altro dei file sorgente; dispone di un certo numero di "Wizard" che svolgono diversi compiti; gestisce anche i file sorgente da elaborare con BibTeX; consente di usare e, volendo, di modificare gli schemi iniziali dei documenti da comporre; consente di creare degli "snippets" per estendere la gestione e il controllo dei comandi e degli ambienti personali allo stesso modo di come esso fa da solo per i comandi e gli ambienti standard di LaTeX e dei pacchetti più comuni; permette di creare degli strumenti di compilazione personalizzati; dispone di un visualizzatore PDF integrato, come TeXworks e TeXShop, e grazie a questo fatto ha lo stesso tipo di sincronizzazione fra il file sorgente e i file composti in formato PDF. Dispone degli strumenti necessari per il controllo ortografico.

La pagina iniziale di questo gedit-LaTeX-plugin si trova in `http://git.gnome.org/gedit-latex`. Ovviamente questo plug-in dipende da altri pacchetti e altre librerie generalmente presenti su ogni macchina Linux; all'occorrenza non è difficile installare le librerie mancanti.

Apparentemente gedit e quindi questo plug-in sono installabili anche sulle macchine Mac con sistema operativo Mac OS X.

TeXworks è lo stesso programma descritto per le macchine Windows, solo che in rete si trovano alcune distribuzioni binarie già usabili e installabili con i soliti strumenti di installazione delle specifiche distribuzioni Linux; altrimenti per tutti è disponibile la "tarball" con i file sorgente ed è possibile compilare il programma con le caratteristiche che si desiderano, ma più propriamente con le funzionalità e le librerie presenti sulla propria macchina.

TeXstudio è lo stesso programma descritto per Windows ed ha le stesse prestazioni oltre alle proprie specifiche, che gli consentono di fare molte più cose e meglio. È importante ricordare che TeXstudio è in grado di interpretare i

commenti di autoconfigurazione di TeXShop, cosa che Texmaker (ancora) non è in grado di fare.

Texmaker è lo stesso programma descritto per le macchine Windows, ma sul suo sito c'è anche un certo numero di eseguibili già predisposti per diverse distribuzioni Linux. Anche con questo programma esiste la possibilità di costruirsi il programma personalizzato per la propria macchina.

Si veda la descrizione svolta nella sezione relativa alle macchine Windows per conoscerne le funzionalità. Molto comodo per usare l'encoding UNICODE, ma questo lo fa anche Kile. I due programmi hanno caratteristiche simili.

4.7.4 Shell editor per le macchine Macintosh

Qui ci si riferisce a macchine Mac operanti con il sistema operativo Mac OS X; questo è un sistema UNIX con l'interfaccia grafica raffinata tipica delle macchine Mac. Su queste macchine possono venire montati anche i programmi validi per Linux e appositamente integrati con l'interfaccia grafica per Mac. Quindi buona parte di ciò che si è detto per le macchine Linux è applicabile anche alle macchine Mac; tuttavia esistono programmi specifici creati e sviluppati per queste macchine; sarebbe opportuno farne uso proprio per sfruttare al meglio le specificità di questo sistema operativo, per il quale ogni documento è generalmente scritto solo usando i formati PDF, oppure RTF, oppure XML (be', molte parti pre-esistenti scritte in HTML sono ugualmente usate, ma generalmente non riguardano i documenti creati dagli utenti).

TeXShop è lo *shell editor* che accompagna di default il sistema TEX, quando questo viene installato con MacTEX. Dal punto di vista dell'editing del testo questo editor non è fortissimo, ma offre tutte le funzionalità principali descritte per gli altri *shell editor*; forse manca di piccole funzionalità di secondaria importanza; esso, dalla versione 2 in avanti è stato molto arricchito di nuove funzionalità e può reggere benissimo il confronto con gli altri *shell editor*. Oggi (2024) la versione corrente è la 5.42; tra le altre cose il visualizzatore integrato e sincronizzato con la finestra di editing, consente di cercare direttamente nel file PDF parole o locuzioni, in modo da poter trovare con un paio di click la locuzione cercata, e per poter passare direttamente alla posizione corrispondente nel file sorgente; questa è una funzionalità preziosissima, specialmente quando si compongono testi di grandi dimensioni. Inoltre questo *shell editor* esegue la verifica ortografica usando le impostazioni che il sistema operativo deduce da quelle di altri programma=i e che il suo programma CocoaSpell riesce a gestire. Sul MAC di chi scrive queste note, è stato impostata di default la lingua italiana, ma anche i driver di tastiera per altre lingue straniere; Ecco allora che TeXShop è in grado di verificare l'ortografia sia dell'italiano, sia dell'inglese sia delle altre lingue che chi scrive potrebbe usare, avendo impostato anche i driver per quelle lingue.

Invece TeXShop possiede nativamente una funzionalità che talvolta manca ancora negli altri *shell editor*, vale a dire la possibilità di navigare nei due sensi fra la finestra di editing testuale e quella di visualizzazione del testo composto. Questa funzionalità è formidabile tutte le volte che si usa il sistema TEX, ma è particolarmente gradita sulle macchine Mac, visto che il formato di default dei documenti composti è comunque il formato PDF. Certo è possibile configurare il motore di composizione affinché il formato di uscita sia quello DVI, ma la funzione di visualizzazione attraverso il programma integrato in TeXShop richiede comunque sempre il formato PDF, per cui l'eventuale file DVI prodotto viene subito "distillato" nel formato PDF. Si può installare un visualizzatore come xdvi per il formato DVI e si può configurare TeXShop in modo che usi quel visualizzatore, invece del suo visualizzatore interno, ma questa sembrerebbe la strada per rendere complicate le cose semplici.

TeXShop è stato il primo *shell editor* a servirsi dei commenti di autoconfigurazione, conosciuti anche con il nome di *righe magiche*. Questi consistono in normali righe di commento all'inizio dell'unico o del primo file sorgente del documento da comporre (in realtà alcune righe magiche sono utili anche nei file secondari) che lo *shell editor* sa interpretare in modo da cambiare temporaneamente la propria configurazione oppure per eseguire azioni particolari; le righe magiche sono le seguenti:

`% !TEX encoding =` ⟨*codifica*⟩
`% !TEX TS-program =` ⟨*programma di composizione*⟩
`% !TEX root =` ⟨*main file completo di estensione*⟩
`% !TeX spellcheck =` ⟨*lingua_varietà*⟩
`% !BIB program =` ⟨*programma di estrazione bibliografica*⟩

dove:

- la ⟨*codifica*⟩ è uno dei nomi delle codifiche usabili dallo *shell editor*, per esempio `Latin1`, `UTF-8 Unicode`, eccetera, ma si consiglia di usare sempre e solo la codifica UTF-8, ricorrendo alle ale solo per documenti datati;
- ⟨*programma di composizione*⟩ può essere uno dei programmi principali per comporre con LATEX, `pdflatex`, `XeLaTeX`, `LuaLaTeX`, eccetera, ma può essere anche un qualsiasi altro programma di composizione del sistema TEX; Un particolare programma di composizione che TeXShop può gestire è latexmk (che richiede il nome `pdflatexmk` per comporre con pdfLATEX, o il nome `lualatexmk` per comporre con LuaLATEX, eccetera). Questo è uno script che controlla se i file in uscita esistono già e se il loro file sorgente sono stati modificati; nel caso esegue tutte le compilazioni sia con il programma di compilazione sia con i programmi di estrazione bibliografici, in modo che i file finali, in particolare il file di uscita, siano completi: ogni volta che si fa una correzione di qualche file sorgente e si ricompila, latexmk provvede a

compilare quante volte occorre finche nei vari file `.log` non ci sono più messaggi di avvertimento. Molto comodo e raccomandabile senza pero esagerare: i tempi di compilazione possono aumentare in proporzione alle ricompilazioni eseguite in questo processo automatico.

- ⟨*main file completo di estensione*⟩ è il nome del file principale per la composizione di un dato documento; se il documento è composto con un solo file sorgente, questa riga magica è inutile, ma non crea problemi se la si usa; essa invece è importante per i vari file inclusi dal main file, perché quando si vuole compilare e si clicca l'apposito bottone dello *shell editor* quando esso è aperto su un file secondario, la compilazione parte con il main file e non con il file secondario il quale, in quanto file incluso, non contiene il preambolo e non può quindi venire compilato autonomamente;
- ⟨*lingua_varietà*⟩ contiene la sigla della lingua seguito dalla sigla della varietà; per esempio il suo valore per l'italiano è semplicemente `it_IT`, mentre per l'inglese potrebbe essere `en_US` o `en_UK` a seconda che per il controllo ortografico si voglia usare il dizionario dell'inglese americano o dell'inglese britannico; grazie alla sua funzionalità automatica del controllo ortografico, questa riga magica è inutile per TeXShop, ma resta utile per il lavoro collaborativo con colleghi che non usano TeXShop;
- ⟨*programma di estrazione bibliografica*⟩ può valere BibTeX oppure biber.

Si noti che specificare il dizionario ortografico con TeXShop non è importante, in quanto questo *shell editor* riconosce da solo quale dizionario usare anche in un documento multilingue; per gli altri *shell editor* in grado di interpretare queste righe magiche, invece, è utile specificare il dizionario ortografico perché non sono in grado di riconoscere da soli la lingua in uso.

Si noti anche che il ⟨*programma di estrazione bibliografica*⟩ preconfigurato per ogni *shell editor* può essere uno solo per ogni documento, anche se lo *shell editor* può essere preconfigurato per usare entrambi; ma quando si clicca il bottone o si sceglie la voce di menù 'Bibliography' viene usato quello dei due programmi specificato nella riga di autoconfigurazione.

Un'ultima osservazione in merito alle righe magiche: la codifica del main file e dei file inclusi deve essere la stessa e deve corrispondere all'opzione specificata al pacchetto *inputenc* quando si compone con pdflatex; con lualatex e xelatex la codifica *deve* essere UTF-8 se si usa anche un solo font OpenType. Non è attraverso i commenti di autoconfigurazione che si può cambiare codifica ad un file sorgente; si può solo adattare lo *shell editor* alla codifica del file sorgente.

Aquamacs è una versione speciale di emacs adattata all'interfaccia grafica dei sistemi Mac; dalla versione 2.0 in poi è particolarmente indicata per il sistema operativo Mac OS X 10.5 e per le versioni successive.

Pur conservando l'intera funzionalità di emacs, contiene una vasta serie di menù con i comandi che normalmente si trovano nei programmi per le piattaforme Mac; inoltre integra già l'estensione *auctex*, per cui quando si crea o si apre un file LaTeX il programma cambia automaticamente il suo aspetto e la sua barra dei comandi per mostrare all'utente quanto è disponibile nel caso si debba editare un file LaTeX. Funziona con lo stesso correttore ortografico di TeXShop. Se il visualizzatore PDF che si associa a Aquamacs è Skim, esso consente la sincronizzazione fra i file sorgente e il file composto in formato PDF più o meno con la stessa funzionalità di TeXworks.

Un suo vantaggio molto utile nella gestione del file sorgente è costituito dalla serie di "interpretazioni" dei comandi LaTeX, che consentono di dare un colore diverso, una forma diversa, una grandezza diversa alle parti di testo marcate con il mark-up di LaTeX; un comando `\section` avrà quindi il suo argomento scritto in corpo decisamente maggiore del testo normale, colorato in blu scuro e sostanzialmente in grassetto. Un testo da evidenziare passandolo come argomento al comando `\emph` apparirà composto nella finestra di editing con un font inclinato e di colore contrastante. Lo stesso succede per le note e per gli argomenti di diversi altri comandi. Questi giochi di colore risultano molto utili durante la fase di creazione e di modifica del file sorgente; va da sé che sono attuati tutti gli altri espedienti utili per evitare gli errori più comuni, per esempio mediante l'accoppiamento delle parentesi dello stesso tipo allo stesso livello.

Esso presenta la possibilità di vedere direttamente nella finestra del file sorgente le equazioni e i disegni già trasformati in immagine composta con pdflatex; è ovviamente possibile tornare al file sorgente anche per correggere questi casi, ma se il risultato è soddisfacente, si può proseguire a elaborare il testo sorgente, e il contenuto della finestra del file sorgente scorre sullo schermo portandosi dietro le parti che hanno già subito questa visualizzazione parziale.

La sincronizzazione fra il file sorgente e il file PDF viene eseguita attraverso un programma di visualizzazione dei file PDF che si chiama Skim; questo è un programma indipendente da Aquamacs e va installato per suo conto; esso può venire configurato per agire in sincronia con diversi editor, ma oggigiorno la maggior parte degli editor moderni dispone di un proprio visualizzatore interno già sincronizzato.

emacs viene distribuito anche per operare nativamente con l'interfaccia grafica del sistema Mac OS X, quindi si può usare la combinazione emacs e auctex per operare come sulle macchine Linux; emacs è molto più potente, come editor testuale, di TeXShop, quindi potrebbe essere una buona scelta; tuttavia, anche con l'aggiunta di auctex, lo *shell editor* che si viene a formare può navigare avanti e indietro solo con i file in formato DVI, e questo porta una pesante ipoteca alla comodità di lavorazione del documento.

Kile Poiché il sistema operativo Mac OS X prevede anche l'interfaccia grafica X Windows (tipica dei sistemi Linux) e consente l'installazione dell'interfaccia KDE, si può usare anche lo *shell editor* Kile; ma sotto certi aspetti può sembrare un regresso rispetto a TeXShop; sotto altri aspetti, invece, Kile può essere migliore; ovviamente dipende dai gusti personali.

TeXworks regge il confronto con TeXShop, sebbene sia ancora un po' giovane. Per aggiornarlo rispetto alla versione impacchettata nell'installatore MacTeX, lo si scarica dal suo sito come file `.zip` che però contiene direttamente l'applicazione `TeXworks.app`; basta estrarla dal file `.zip` e metterla nella propria cartella `Applications`, per averlo direttamente funzionante. Ci si ricordi che per impostare la configurazione desiderata, dopo averlo lanciato almeno una volta, anche senza aprire nessun file, bisogna fare come al solito con i programmi Mac, cioè aprire il menù a discesa cliccando sul nome del programma presente nella barra superiore e lì scegliere "Preferences". Nei programmi per Windows e per Linux, invece la scelta delle preferenze personali si trova come ultimo elemento del menù a discesa della voce "Edit" ("Modifica" quando il programma è localizzato in italiano).

Benché questo programma nasca da TeXShop, presenta alcune funzionalità assenti in TeXShop; altre funzionalità sono eseguite in modo diverso, come per esempio la creazione del legame di subordinazione di un file sorgente al suo master file. Così la configurazione dei programmi, delle scorciatoie, della codifica, eccetera. È comodo che il pulsante per eseguire la composizione sia presente tanto nella finestra del file sorgente quanto nella finestra del file composto.

Più avanti si vedrà come usare TeXworks per convertire la codifica dei file; anche altri editor consentono di farlo, (aprire un file codificato in un modo e salvarlo codificato in un altro) ma come lo si fa con TeXworks è particolarmente comodo.

TeXstudio ha gli stessi pregi già descritti come *shell editor* multipiattaforma; dispone del suo visualizzatore interno per file PDF già sincronizzato con il file sorgente. Esso, dalla versione 2.4, è anche in grado di interpretare le righe di autoconfigurazione alla TeXShop, cosa che risulta molto comoda in assoluto, ma anche per agevolare il lavoro cooperativo fra autori che usano macchine e sistemi operativi diversi.

Texmaker presenta le stesse funzionalità descritte per i sistemi Windows, e le stesse limitazioni. Nonostante ciò è un ottimo shell-editor; esso è adatto anche alle macchine Mac e contiene un suo visualizzatore interno predisposto per la sincronizzazione.

4.8 Editor quasi WYSIWYG

Sembra opportuno segnalare che esistono dei programmi gratuiti o commerciali che permettono di lavorare i file sorgente `.tex` in maniera quasi WYSIWYG; se ne sconsiglia l'uso perché al fine di presentare la composizione del documento più o meno con la qualità di LaTeX devono rinunciare a molte funzionalità.

Il programma TeXmacs mostra sullo schermo il risultato della composizione proprio come lo farebbe LaTeX, usando persino gli stessi font di default di LaTeX. Non permette di cambiare font né di usare qualunque pacchetto di estensione; il programma è gratuito, ma appunto è estremamente limitato. Sembra che le ultime versioni siano un po' meno limitate; al suo interno non viene usato LaTeX, ma il programma è in grado di importare file LaTeX con alcune limitazioni, e di esportare i suoi file in formato LaTeX, completi di preambolo e di corpo del documento. Come tutti i file che si dichiarano WYSIWYG è in grado di fare solo quello per cui è stato programmato. Può essere interessante usarlo una volta, ma poi ci si stanca delle sue limitazioni. L'utente che si avvicina a LaTeX attraverso TeXmacs s'accorge ben presto delle sue limitazioni e cessa di usarlo.

Il programma LyX non presenta nella schermata esattamente quello che verrà prodotto con LaTeX, ma ne mostra un buona approssimazione; consente di usare i pacchetti di estensione, quindi in teoria permette di usare LaTeX in tutte le sue sfaccettature. In realtà non è così; spesso consente di caricare pacchetti di estensione, ma per "disegnare" la schermata con l'approssimazione del risultato della composizione con LaTeX, ha bisogno di appositi file di interfaccia che "traducano" il mark up LaTeX nel linguaggio interno di LyX. Il sistema TeX dispone di alcune migliaia di pacchetti, formati spessissimo da molti file; generalmente chi crea questi pacchetti non usa LyX e quindi non produce i file di interfaccia; come conseguenza usare LyX serve a poco o niente, anzi complica le cose per inserire pacchetti e comandi "nascosti" che disturberebbero la schermata dalle pretese di essere praticamente il risultato finale della composizione con LaTeX. Non è così, ma il suo canto di sirena attrae molti utenti che poi si accorgono troppo tardi che non ne valeva la pena.

4.9 LaTeX e i programmi di composizione del sistema TeX

Come si è già segnalato nelle Avvertenze, è bene che il lettore sappia che quando si dice LaTeX si intende il linguaggio di mark-up specifico per descrivere le componenti di un documento e di cui parleremo approfonditamente nel resto di questa guida. Nel linguaggio corrente si usa lo stesso nome per indicare il programma che interpreta il mark-up ed esegue la composizione del documento. Quindi esso indicherà non solo il programma latex vero e proprio, ma anche i programmi pdflatex, lualatex e xelatex. Questi ultimi hanno la caratteristica che il loro formato di uscita è direttamente il formato PDF e l'utente non ha bisogno di intervenire personalmente per convertire i file di uscita nel formato PDF.

Vale la pena sottolineare immediatamente le differenze nel file finale che si ottiene direttamente con pdflatex o lualatex rispetto a quello che si può ottenere mediante le trasformazioni già menzionate, cioè eseguendo in sequenza i programmi latex, dvips e ps2pdf, oppure latex e dvipdfm.

I programmi pdflatex e lualatex producono un file PDF conforme agli standard e dotato di tutte le caratteristiche che gli standard consentono, dalle annotazioni ai *thumbnail*[9], dai link ai file o siti esterni, ai link di navigazione interna del documento. La composizione diretta in formato PDF consente anche una qualità tipografica decisamente superiore grazie alla possibilità di usare la microtipografia; se ne farà una descrizione abbastanza ampia in un successivo capitolo. Apparentemente xelatex produce direttamente il file di uscita in formato PDF, ma in realtà esso è un'(importante) estensione del programma latex, ma è concepito in modo tale da eseguire da solo la conversione dal formato DVI (esteso) al formato PDF.

Passare attraverso il formato DVI, quello di default che si produce con il programma latex, comunque lo si trasformi, non consente di ottenere tutte le funzioni del formato PDF 'vero', perché il formato DVI, di sua natura, non può contenere certe informazioni. Esso è un modo per descrivere il layout di una pagina tipografica, non di descrivere gli oggetti che ne fanno parte. È vero, esiste la possibilità di aggirare queste limitazioni mediante un comando speciale che, appunto, si chiama `\special`, e che permette di inserire nel file di uscita dei brani di un altro linguaggio che verrà poi interpretato (o ignorato, se non è conforme) dal programma di 'traduzione' del formato DVI in un altro formato oppure per essere visualizzato sullo schermo o per essere stampato. Va comunque precisato che il formato DVI non consente l'uso della microtipografia, e quindi, benché la sua qualità tipografica sia molto buona, non è altrettanto buona di quella che si può ottenere con la generazione diretta del file PDF attraverso i programmi pdflatex e lualatex.

Il formato PDF si chiama appunto Portable Document Format, proprio perché consente di 'portare' i documenti composti su qualunque piattaforma dotata di qualunque sistema operativo, basta che disponga di un visualizzatore di file PDF; e perciò è particolarmente indicato per i *libri elettronici* (o *e-book* come spesso vengono chiamati oggi) previsti non per essere stampati ma per essere 'sfogliati' sullo schermo. Ovviamente il formato PDF va benissimo anche per la stampa ma, una volta stampato, le possibilità di navigazione interna o esterna vengono comunque perse.

Fino a poco tempo fa, e in certi ambienti ancora oggi, veniva data una grossa preferenza al formato PostScript; chi lavora in ambiente UNIX (e Linux) continua

[9]La parola inglese *thumbnail* alla lettera significa *unghia del pollice*; in italiano si userebbe la parola *unghia*, ma si riferisce a quegli intagli semicircolari sul margine esterno delle pagine che si trovano in grossi volumi di consultazione, come dizionari ed enciclopedie, e che servono per agevolare la ricerca di una particolare sezione alfabetica del volume. In informatica si attribuisce il nome di *thumbnail* alle immagini ridotte delle pagine o delle figure che compaiono spesso sui bordi dei programmi di visualizzazione dei file PDF o nei programmi di sistema che consentono di cercare i file nelle cartelle dei dischi.

a privilegiare il formato PS grazie al fatto che questo formato negli anni '80 e nei primi anni '90 era l'unico che potesse ricorrere a stampanti per la stampa diretta del formato PostScript. Oggi i libri elettronici sono relativamente diffusi e stanno diffondendosi ancor di più con la messa in commercio di piccoli apparecchi capaci di funzionare non solo come telefoni cellulari, ma anche come navigatori di rete, lettori di libri elettronici, riproduttori di file audio, eccetera. Dal punto di vista archivistico oggi esiste una norma ISO che impone l'uso del formato PDF al fine di produrre documenti elettronici archiviabili a lungo termine.

Nel seguito, quindi, ci si riferirà quasi sempre al formato PDF prodotto direttamente con i programmi pdflatex e lualatex.

I programmi pdflatex, lualatex e xelatex, che contengono la stringa latex nel loro nome, sono in grado di interpretare le istruzioni del linguaggio LaTeX grazie ad un file poco conosciuto, di solito "trasparente" agli occhi dell'utente normale, che si chiama *file di formato*.

Il file di formato è un file che contiene una collezione di definizioni di macroistruzioni che vengono lette come prima cosa dal vero motore di composizione; siccome queste macroistruzioni vengono lette sempre, allora il sistema TeX, fin dall'inizio, ha sviluppato la tecnica di raccogliere le definizioni comuni legate al tipo di mark-up in file speciali, per LaTeX nel file `latex.ltx`, per `plain` TeX nel file `plain.tex`, eccetera; questa tecnica consente di leggere una volta per tutte questi file contenenti le macro comuni mediante una versione 'vergine' dei programmi tex oppure pdftex oppure xetex oppure luatex, che, dopo avere letto i file e averli elaborati per trasformarli in linguaggio macchina, li memorizzano sul disco di sistema in file binari con estensione `.fmt`. Quando si lancia uno dei programmi di composizione, in realtà si lancia un programmino di interfaccia, che a sua volta lancia il programma interprete dicendogli anche di usare il file di formato `latex.fmt` o gli altri analoghi file di formato predisposti per gli altri programmi.

Questa delucidazione può sembrare capziosa, specialmente se si aggiunge che dal 2005 ogni distribuzione del sistema TeX contiene anche i programmi interpreti estesi (vedi più avanti il paragrafo 4.10.2); questi sono retrocompatibili con i programmi precedenti, per cui li sostituiscono completamente dal punto di vista funzionale.

Per complicare le cose al neofita, che oltretutto, almeno all'inizio, difficilmente si accorgerà delle differenze, bisogna aggiungere che il programma pdfetex, specificando un opportuno parametro, si comporta in modo esattamente uguale a etex. Dal 2007 pdftex è diventato il programma di composizione che congloba sia l'interprete etex sia pdfetex, e questi ultimi due non vengono citati se non molto raramente; l'utente non se ne accorge, ma per chi si dedica alla manutenzione dei programmi la cosa rappresenta una enorme semplificazione.

Per tradizione l'interprete tex, benché le sue funzionalità siano un sottoinsieme di quelle di pdftex, continua ad essere incluso in ogni distribuzione del sistema TeX. Se un altro programma, comunque si chiami, svolge almeno tutte le funzioni del programma tex, esso ha diritto di chiamarsi o di far parte della collezione della distribuzione del sistema TeX. Questo programma, progenitore di tutti

gli altri, continua ad essere la pietra di riferimento per tutto il sistema; oggi non viene quasi più usato, se non, appunto come pietra di paragone. Non va dimenticato che il suo creatore, Donald E. Knuth, l'ha congelato e non intende più modificarlo; si limita a correggere i pochi errori che ancora contiene, premiando con un assegno firmato di suo pugno chiunque gli segnali un vero errore. Non va in bancarotta per due motivi: (*a*) di errori ce ne sono pochissimi, e (*b*) chiunque riceva un assegno firmato da Knuth stesso, non lo incassa ma lo espone in cornice. Il numero di versione di tex contiene le prime cifre del numero π, e ad ogni modifica acquisisce una nuova cifra tendendo asintoticamente a π, cosa che non avverrà mai visto che π è trascendente. Oggi il numero di versione è 3,14159265; visto che le poche correzioni a tex vengono immediatamente inserite anche in pdftex, questo numero lo si ritrova nella prima riga di ogni file `.log`, qualunque sia il programma effettivamente usato direttamente o indirettamente attraverso lo *shell editor*.

Ciò non toglie che per pdftex esistano due modalità di composizione (non importa se sono due distinte modalità di funzionamento dell'unico programma: pdftex): quella che in uscita produce un file DVI e quella che produce invece un file PDF.

I file sorgente elaborabili con latex o con pdflatex, xelatex o lualatex sono in generale identici, o possono essere resi tali, nel senso che usano lo stesso mark-up LaTeX. Per cui quando si parla di un file elaborabile con latex, si intende che è stato scritto facendo uso del metodo di mark-up tipico di LaTeX, e che il file è elaborabile indifferentemente anche da pdflatex, xelatex o lualatex. Sarebbe desiderabile che i file sorgente con questo mark-up avessero l'estensione `.ltx`, invece che l'estensione `.tex`, ma per vari motivi questa convenzione viene rispettata solo per i file originali distribuiti con TeX Live e MikTeX.

A causa dell'uso di pacchetti di estensione molto specializzati, un file apparentemente conforme alla grammatica del mark-up di LaTeX può essere elaborato con latex, ma può risultare incompatibile per essere elaborato con pdflatex o xelatex o lualatex; può succedere anche il contrario. Ciò succede raramente con pdflatex, ma non è impossibile che accada, ma succede quasi sempre con xelatex e lualatex; ciò dipende principalmente dal fatto che questi ultimi due programmi sono estensioni molto avanzate rispetto a pdflatex e possono gestire le lingue e i font in modi che escludono pdflatex dal confronto; lualatex, inoltre incorpora un programma di scripting, con il quale si possono fare cose che con il solo linguaggio nativo di pdftex sono impensabili.

Talvolta l'incompatibilità fra latex e gli altri programmi di composizione non dipende dal file sorgente, ma dai file inclusi; ciò può accadere facilmente con i file che contengono delle immagini; è per questo che sono molto utili quei programmi accessori che consentono di trasformare un'immagine da un formato all'altro; a questo proposito si faccia riferimento al capitolo 10.

La tabella 4.1 riassume in parte quanto è stato esposto. L'unica differenza, vista la moltitudine di formati da generare in pratica con un numero inferiore di interpreti, è che invece di usare dei file sorgente con estensione `.tex` o `.ltx`, si usano dei file di inizializzazione con estensione `.ini` che provvedono a far leggere

Comando di sistema	Interprete	File di inizializzazione	File di formato compilato
tex	tex	`tex.ini`	`tex.fmt`
etex	pdftex	`etex.ini`	`etex.fmt`
pdftex	pdftex	`pdfetex.ini`	`pdftex.fmt`
eplain	pdftex	`eplain.ini`	`eplain.fmt`
latex	pdftex	`latex.ini`	`latex.fmt`
pdflatex	pdftex	`pdflatex.ini`	`pdflatex.fmt`
xelatex	xetex	`xelatex.ini`	`xelatex.fmt`
lualatex	luatex	`lualatex.ini`	`lualatex.fmt‘`

Tabella 4.1: Legame fra alcuni comandi di sistema, gli interpreti e i formati

all'interprete i file specifici per il formato e per impostare i parametri necessari all'interprete per produrre i formati giusti in relazione al particolare modo di funzionamento del programma di sistema che si desidera attivare.

In pratica i primi quattro formati attraverso i file di inizializzazione leggono tutti il file `plain.tex` insieme ad altri file con possibili estensioni del linguaggio, e impostano i parametri interni dell'interprete affinché l'esecuzione dei programmi i cui nomi *non cominciano* con pdf producano l'uscita in formato DVI, mentre quelli il cui nomi *cominciano* con pdf producano l'uscita in formato PDF. I successivi due si riferiscono al programma latex la cui uscita è in formato DVI e al programma pdflatex la cui uscita è in formato PDF. Entrambi i formati sono creati leggendo sostanzialmente lo stesso file `latex.ltx` ma con inizializzazioni diverse a seconda del formato di uscita di default. Infine gli ultimi due casi si riferiscono ai programmi estesi xelatex e lualatex di cui si dirà di più qui di seguito.

4.10 Altri programmi del sistema TEX

Vale la pena ricordare in questo capitolo la presenza di altri programmi del sistema TEX. Essi hanno campi di applicazione in parte complementari a quelli di LATEX e spesso vengono installati di default quando si installa il sistema TEX.

Tuttavia questo paragrafo e il successivo possono essere tranquillamente letti in un secondo tempo; servono per capire quanto sia vasto il mondo di TEX, ma costituiscono più una curiosità che non un insieme di informazioni utili a capire, sia pure a livello introduttivo, la potenza della tipografia con LATEX.

4.10.1 Plain TEX

Il mark-up semplice, 'plain' in inglese, usato nei primi quattro esempi della tabella 4.1 è quello formato dall'insieme di comandi predisposto dall'autore Knuth e con il quale, tra l'altro, egli continua a comporre tutti i suoi testi.

Come linguaggio di mark-up è molto meno strutturato di LaTeX, tanto che talvolta il suo insieme di comandi viene qualificato come 'prescrittivo', invece che 'descrittivo'; LaTeX ha un insieme di comandi di mark-up decisamente più descrittivo, ma non è completamente descrittivo; così il mark-up Plain ha un insieme di comandi di gran lunga molto prescrittivo, e solo qualche comando di tipo descrittivo.

I programmi che usano il mark-up Plain, semplice o esteso[10], con uscita DVI o PDF, oggi vengono usati poco, ma in qualche raro caso si incontrano dei file sorgente scritti da altri che possono essere composti ricorrendo a tex o a pdftex. In questo caso gli *shell editor* sono in grado di eseguire questo compito con la semplice pressione di uno dei vari pulsanti presenti nei vari menù, quindi l'utente 'normale' non ha bisogno di conoscere il mark-up Plain. Sappia tuttavia che l'insieme di comandi di LaTeX costituisce una enorme sovrastruttura sui comandi di Plain TeX e che quasi tutti i comandi di Plain sono usabili sotto LaTeX e generalmente producono lo stesso effetto.

Per la documentazione ci si può rivolgere direttamente alla pietra miliare costituita dal libro e manuale di Knuth, il TeXbook, [35], che non dovrebbe mancare dalla scrivania di qualunque utente avanzato di LaTeX.

4.10.2 I programmi estesi

I programmi etex, elatex, pdfetex e pdfelatex deriverebbero dalla generazione del formato mediante il programma pdfetex; questo oggi è totalmente assorbito dentro pdftex, quindi quei programmi non esistono più; tuttavia se ne parla solo per descrivere le estensioni di etex introdotte in pdftex.

Come si vede dalla tabella 4.1 oggi tutti questi programmi sono prodotti creando il formato sempre con l'interprete pdftex e quindi sostituiscono completamente tutti i programmi con la 'e' citati sopra; chi usa i programmi della tabella usa sempre programmi estesi, anche se non usa nessuna delle estensioni.

Insomma i programmi di oggi sono *downward compatible* con i rispettivi progenitori. Tutti i pacchetti di estensione che fino ad oggi sono stati sviluppati per TeX, LaTeX e i loro 'fratelli' sono quindi usabili anche con i programmi estesi; invece a tutto il 2014 erano ancora pochi i pacchetti di estensione che richiedevano espressamente l'uso dei programmi estesi e non erano compatibili con i progenitori, ma stanno aumentando di numero rapidamente. Inoltre dal 2018-2019 il LaTeX Project Team sta introducendo in LaTeX molte funzionalità nuove oltre a correggere alcuni (rari) difetti che si sono via via manifestati nei successivi aggiornamenti del codice; a questo proposito si veda il capitolo 30.

C'è da sperare, però, che tutti i pacchetti di estensione esistenti vengano via via aggiornati, non tanto per dotarli di nuove funzionalità, quanto per sfruttare le estensioni dei 'modi' di esecuzione che possono svolgere nativamente certe

[10] Sono estesi i mark-up che contengono la lettera 'e' prima di 'plain' o prima di 'tex'; l'estensione consiste nell'uso della sillabazione per molte lingue (valido solo per il mark-up discendente da `plain.tex`) e nella disponibilità dei comandi estesi descritti successivamente in merito all'interprete etex.

funzioni, che fino ad oggi non sono sfruttate oppure che sono simulate con sequenze di istruzioni TEX o LATEX.

Vale la pena di sottolineare un fatto che a prima vista potrebbe apparire come superfluo, ma non lo è; è vero che fino ad oggi i programmi generalmente distribuiti su CTAN non sfruttano le nuove potenzialità o funzionalità dei programmi estesi, anche se questi sono i motori veri di molte varianti dei programmi di composizione del sistema TEX.

Ma una cosa è quella di creare i file di formato che sono sinonimi dei 'programmi' da lanciare; un'altra è quella di sapere in quale modalità sta operando il motore di composizione; finora questo motore di composizione è stato chiamato *interprete*, perché il suo scopo è quello di interpretare le definizioni delle macroistruzioni per tradurle in linguaggio macchina, al fine di produrre un file di uscita contenente il testo tipocomposto.

Se con questo interprete si generano i file di formato avendo specificato `\pdfoutput=0`, il lavoro di composizione si concluderà con un file di uscita in formato DVI; al contrario se viene specificato `\pdfoutput=1`, il lavoro di composizione termina con un file in formato PDF. Il valore specificato per il parametro `\pdfoutput` viene integrato dentro il file di formato, per cui l'esito della composizione è predefinito, a meno che il compositore non voglia egli stesso specificare all'inizio del suo file sorgente un diverso valore del parametro `\pdfoutput`.

Ognuno può esaminare il proprio *shell editor* per vedere quale programma viene lanciato quando si clicca sull'icona che rappresenta LATEX. Per chi scrive, che usa TeXShop su una macchina Mac, il bottone "Typeset" corrisponde a lanciare l'interprete pdftex in modalità estesa con il file di formato prodotto con `\pdfoutput=1`, quindi egli produce di default, in modalità estesa, file di uscita in formato PDF. Con lo shell editor LED su una macchina Windows, invece, il bottone "LATEX" lavora con pdftex in modalità estesa ma produce in uscita un file DVI perché il formato `latex.fmt` è stato creato con `\pdfoutput=0`; quando si preme il bottone "pdfLATEX" si lancia lo stesso interprete ma si lavora con un altro formato creato con `\pdfoutput=1`.

Generalmente oggi i file di formato sono tutti o quasi tutti composti in modalità estesa, o lo si può fare come si è descritto sopra. Tuttavia per trarne il miglior beneficio bisogna documentarsi a fondo leggendo la documentazione con `texdoc etex`.

Ma in che cosa consiste questa modalità estesa? Permette di superare la limitazione di 256 contatori, scatole, lunghezze rigide, lunghezze elastiche, eccetera, che erano uno dei limiti del programma tex. Permette di comporre il testo sia con alfabeti 'normali' (da sinistra a destra) sia con alfabeti retrogradi (da destra a sinistra). Permette di eseguire alcuni semplici calcoli con i contatori e le lunghezze direttamente all'interno di espressioni speciali, ma facendo uso, sostanzialmente delle stesse notazioni operazionali che siamo soliti usare quando scriviamo l'algebra. Permette di eseguire alcuni altri test non previsti da tex, come per esempio verificare se un particolare segno esiste in un determinato font. Si può approfondire nella documentazione indicata sopra, anche se si riferisce

ad un programma che non esiste più, ma le cui funzionalità sono state trasfuse tutte in pdftex.

Secondo lo scrivente uno dei vantaggi della modalità estesa consiste anche nella possibilità di lanciare altri programmi durante l'esecuzione della composizione e questo, per esempio, consente di convertire al volo il formato delle immagini come si descriverà meglio nel capitolo 10 e, specialmente, per gestire gli indici analitici come esposto nel paragrafo 12.5.

I programmi di composizione xelatex e lualatex lavorano sempre in modalità estesa; grazie a ciò possono eseguire certi programmi esterni che permettono loro di superare alcune piccole limitazioni dei programmi latex e pdftex.

4.10.3 Il mark-up ConTEXt

Il mark-up ConTEXt assomiglia molto vagamente sia a Plain, sia a LATEX. Esso è un mark-up molto strutturato, ma tutto sommato semplice come Plain; la differenza sta nel fatto che virtualmente ogni comando accetta che vengano specificate delle opzioni per cui la personalizzazione della composizione avviene in modo più semplice rispetto a quanto sarebbe necessario fare con Plain.

Esso permette di descrivere il testo da comporre in modo molto più strutturato anche rispetto a LATEX, e i libri e gli altri documenti composti con questo programma sono generalmente molto ben riconoscibili per l'altissima qualità del risultato finale. Tutto ciò avviene grazie alle opzioni previste e attivate dai vari comandi usati per la descrizione del testo. Gli autori hanno spesso affermato che essi preferiscono la libertà di definire qualunque dettaglio compositivo ma non volevano perdere le funzionalità offerte da LATEX; direi che ci sono riusciti magnificamente e i risultati si vedono. Tuttavia il mark-up è così differente da quello di LATEX che la maggior parte degli utenti continua a usare LATEX.

ConTEXt inizialmente aveva come motore di composizione pdftex e quindi conteneva le sue estensioni e i suoi limiti; il principale è quello di essere limitato a font che non contengano più di 256 caratteri, quindi non può lavorare con i font OpenType codificati in UNICODE. Questa versione di ConTEXt si chiama ConTEXt Mk II.

Dal 2013 è operativa una versione chiamata ConTEXt Mk IV che supera questi limiti; questa nuova versione si appoggia all'interprete luatex, e a quello si rimanda per ulteriori informazioni.

4.10.4 I programmi Omega e Lambda, Aleph e Lamed

Il programma Omega, il cui sviluppo è ora arrestato, cerca di sostituire il programma TEX in modo da togliergli tutte le limitazioni che ha, per esempio quella di trattare font con non più di 256 segni, e di arricchirlo di altre funzionalità. Tra le altre funzioni consente anche la composizione con alfabeti retrogradi; con l'arabo, che è sostanzialmente ancora una scrittura calligrafica molto legata, consente di eseguire legature molto complesse che coinvolgono anche quattro o cinque segni. In altre parole è decisamente più esteso di pdftex.

Il programma Lambda è, per così dire, la versione LaTeX di Omega; ne accetta tutte le estensioni e mette in grado il compositore di riferirsi ad una sintassi simile a quella di LaTeX.

Il programma Aleph è una versione estesa di Omega che permette di liberarsi di certi vincoli sui font usabili; per un certo periodo anche ConTeXt poteva usare Aleph come motore di composizione. L'autore di Aleph, Giuseppe Bilotta, ha lavorato anche al programma Lamed, versione di Aleph LaTeX-compatibile. Sembra però che anche questo programma sia stato assorbito in parte da luatex, e quindi il suo sviluppo ulteriore sia stato interrotto. Non deve però essere sottovalutata l'importanza di Aleph, perché rappresenta il passaggio da pdftex a luatex. Molte sue librerie sono state infatti incorporate in luatex.

4.10.5 Il programma XeTeX

Il programma XeTeX[11], ora disponibile per tutte le piattaforme di elaborazione e con qualunque sistema operativo, è una ulteriore estensione che riunisce molte delle funzionalità dei programmi precedentemente descritti, ma in più consente di usare tutti i font presenti sul particolare calcolatore dove può essere impiegato, purché questi siano font di tipo OpenType, oppure TrueType oppure PostScript. Non occorrono estensioni particolari per l'uso di questi font, perché il programma sfrutta i comandi del sistema operativo per gestire i font disponibili e quindi una delle operazioni relativamente meno semplici da fare con LaTeX per gestire i font viene superata in un solo colpo. Il programma sta diffondendosi con rapidità specialmente in certi ambiti disciplinari; si tenga presente che gestisce tutti i possibili font di qualunque alfabeto, anche i sistemi di ideogrammi e gli alfabeti retrogradi come quello ebraico o quello arabo. Mescolare lingue diverse scritte con sistemi di caratteri completamente diversi diventa quindi una operazione semplicissima.

Ovviamente, per raggiungere questi scopi deve poter usare sia in entrata, sia in uscita la codifica UNICODE, anche se ha delle funzioni che gli consentono di usare i font vettoriali del sistema TeX, benché non siano codificati in UNICODE in senso stretto.

Esiste anche la versione LaTeX di XeTeX, chiamata XeLaTeX, e non è particolarmente difficile usarla pur di riferirsi al particolare manuale che l'accompagna, se non altro per prendere nota delle differenze/estensioni rispetto al LaTeX standard.

Il passaggio dalla compilazione con LaTeX a quella con XeLaTeX è praticamente indolore; se non si richiamano font diversi da quelli standard del sistema TeX, lo stesso file sorgente può venire composto sia con pdflatex sia con xelatex e si

[11] XeTeX è scritto in modo palindromo (simmetrico rispetto alla mezzeria) e lascerebbe pensare che si debba pronunciare anche in modo palindromo; il suo creatore, Jonathan Kew, ha generato il nome pensando a *eXtended ε-TEX* e quindi la prima 'X' non è una 'χ' maiuscola, ma è davvero una 'X'. Il nome dovrebbe quindi essere pronunciato 'xetekh'; il suo creatore lo pronuncia all'inglese 'zitek', ma a differenza di D. Knuth, non ne impone una pronuncia particolare.

ottengono gli stessi risultati. I font usati di default sono quelli della collezione Latin Modern; infatti, invocando il pacchetto *fontspec* senza specificare nessun font particolare, viene usata la versione OpenType dei font che fanno parte della collezione Latin Modern. Si può usare anche il pacchetto *unicode-math* per usare le particolarità delle polizze di font che contengono anche i segni matematici codificati con la codifica UNICODE e si dispone di una scelta sterminata di segni matematici (oltre ad una miriade di altri segni) che possono mettere nell'imbarazzo della scelta. Anche altri programmi di impaginazione possono accedere ai font matematici UNICODE, ma solo il motore xetex riesce a gestirli in modo impeccabile rispetto a qualunque altro impaginatore o word processor.

4.10.6 Il programma LuaTEX

Il programma luatex è già nato, e nel 2016 ha raggiunto la versione stabile 1.00.0; perciò luatex è già abbondantemente usabile oggi, ma in condizioni controllate, perché è ancora soggetto ad una certa evoluzione. Questa volta si tratta veramente di un nuovo programma, che accoglie in sé diverse librerie C di pdftex (che a sua volta contiene sia tex sia etex), diverse di Aleph (che già contiene Omega), diverse di METAPOST, diverse di kpathsea, diverse del linguaggio di scripting lua, alcune delle quali sono state scritte apposta, diverse funzionalità del visualizzatore xpdf, alcune dell'editor di font FontForge.

Il file sorgente di un documento da comporre può seguire il mark-up di ConTEXt mk IV e contiene sia il testo da comporre, sia le istruzioni per comporlo, ma l'interprete luatex è in grado di svolgere anche altre funzioni che, per esempio, l'interprete pdftex non è capace di eseguire. Per esempio, gli interpreti classici del sistema TEX non erano in grado di eseguire agevolmente calcoli con operandi in notazione *floating point*[12] tipica di ogni programma di calcolo; per questi interpreti un numero scritto nella forma `-0.12345E02` era privo di senso; pdftex è in grado di usare il numero decimale `-12.345` (equivalente a quello indicato in notazione floating point) solo come fattore di scala per le lunghezze. Per poter eseguire calcoli di ogni genere con numeri decimali e/o floating point bisognava ricorrere a macro specializzate con pdftex, mentre con luatex i calcoli si possono fare tranquillamente. Ciò è possibile per il fatto che luatex contiene un certo numero di librerie lua, compresa la libreria matematica, cosicché con i comandi di lua eseguibili all'interno di luatex si possono generare valori floating point già frutto di elaborazioni, e passare questi numeri ai moduli di composizione di luatex per usarli nel modo che il sottostante linguaggio PDF possa gestirli nel modo richiesto per l'output.

Ora luatex e lualatex (ma anche pdfLATEX e XELATEX tramite il pacchetto *xfp*) possono usare la libreria del linguaggio LATEX 3, `l3fp`, che permette di usare completamente la matematica decimale in virgola mobile e in notazione scientifica.

[12] Ora con lo sviluppo del linguaggio LATEX 3 anche i programmi tradizionali basati su LATEX possono eseguire calcoli in notazione *floating point* tramite il pacchetto *xfp*.

Il pacchetto di interfaccia per l'utente, *xfp* permette all'utente di usare macro compatibili con LaTeX per eseguire i calcoli con questa matematica decimale. La libreria è abbastanza ricca, ma l'interfaccia *xfp* (disponibile dal 2018) ancora non permette di usarla completamente. Tuttavia è ancora in sviluppo e certe limitazioni attuali verrano rapidamente superate.

Questo stesso principio vale per ogni altra caratteristica del testo da comporre visto che le librerie di lua sono in grado di eseguire ogni operazione eseguibile da un calcolatore. Perciò luatex è in grado di elaborare file da comporre con alfabeti strani, di eseguire ogni sorta di comandi condizionali in modo che lo stesso file sorgente possa produrre l'uscita in diversi formati, possa gestire i font OpentType testuali e matematici in modo relativamente semplice, eccetera. Una delle cose difficili da fare con pdftex, se non impossibili, ma facili da fare con luatex è quella di caricare i pattern di sillabazione al momento dell'esecuzione; un'altra è quella di generare al volo font virtuali e di crearne le relative informazioni metriche da usare durante la stessa esecuzione del programma. Grazie alle librerie di Aleph è possibile gestire le complicatissime legature, che possono coinvolgere anche quattro o cinque glifi, con alfabeti come l'arabo, che è molto calligrafico, ma anche come il font Zapfino, uno script "latino" fortemente calligrafico come appare nella figura 2.2.

Informazioni dettagliate si possono trovare nel sito del progetto LuaTeX: `http://www.luatex.org/`. Siccome è già stata presentata le versione 1.00.0, c'è da aspettarsi che LuaTeX diventi presto la soluzione di default per tutti o quasi i compiti di composizione tipografica. Un bel documento che traccia la storia dello sviluppo di ConTeXt e illustra le potenzialità di luatex si può scaricare da `http://www.pragma-ade.com/general/manuals/mk.pdf`.

Tuttavia non dimentichiamo che luatex non serve solo per lavorare con il mark up ConTeXt, ma serve anche per LaTeX; infatti si dispone del programma lualatex che, appunto, contiene tutte le definizione del linguaggio di programmazione LaTeX di cui ci occupiamo in questo testo.

Dal 2013 un file sorgente contenente un preambolo che sappia riconoscere quale sia il programma che si sta usando per comporlo e usi un preambolo conforme con quel programma, può venire compilato con i tre programmi principali descritti sopra se non si ricorre a funzioni speciali sui font e, soprattutto, senza ricorrere al linguaggio Lua. I tempi di esecuzione sono diversi; il più veloce è pdflatex e il più lento è lualatex, con xelatex che si colloca in posizione intermedia. Vale la pena di notare che pdflatex e lualatex permettono di usare la microgiustificazione completa, mentre xelatex ne può sfruttare solo la *protrusione* nei margini. Perciò il risultato della composizione è leggermente diverso nei tre casi, anche perché lualatex e xelatex possono usare i font OpenType, mentre pdflatex di fatto usa solo i font Type 1. Quindi, quale dei tre programmi di composizione usare? La domanda non ha una risposta definitiva: dipende evidentemente da ciò che si vuole comporre.

4.11 Il sistema TEX

A conclusione di questo capitolo si spiega perché in questo testo si parla di "sistema TEX". Ormai dovrebbe essere chiaro, ma vale la pena di scendere in ulteriori dettagli.

Un sistema è un insieme coordinato di programmi che permettono di eseguire operazioni semplici o complesse in un ambito vasto. Di solito non sono composti di un unico programma o applicazione o eseguibile, comunque lo si voglia chiamare. Di solito è modulare in modo che per ogni operazione siano usate solo le risorse che sono utili, cioè solo i moduli funzionali per quella operazione. Questo è molto più efficiente di un unico immenso programma che faccia tutto, ed è molto più facile da rivedere per eliminare gli inevitabili piccoli errori o per aumentare le funzionalità complessive.

Il lettore curioso può esplorare il proprio disco rigido, dove ha installato una distribuzione del sistema TEX; troverà una cartella `bin/`⟨*piattaforma*⟩`/` che contiene i programmi direttamente eseguibili dalla macchina corrispondente alla particolare ⟨*piattaforma*⟩, oppure degli script eseguibili attraverso programmi di interpretazione, quali Perl, Java, Python, Bash, e simili; le macchine Windows interpretano a livello di sistema operativo le procedure *batch* scritte in file con l'estensione `.bat`.

Sulla macchina che chi scrive sta usando, un Mac con la distribuzione MacTEX 2018, la cartella `bin/x86_64-darwin/` contiene i nomi di circa 450 file, costituenti altrettante funzionalità del sistema nel suo complesso. Sulle macchine Windows e Linux il numero di eseguibili e di script può essere maggiore o minore, ma non è molto diverso.

Non si vogliono elencare qui le funzionalità di ciascuno di questi 450 programmi; ma si citeranno quei pochi che l'utente si può provare ad usare in prima persona; in ogni caso quei programmi così importanti di cui non si può ignorare l'esistenza. Tutti quelli che si elencano qui sono già stati nominati o verranno nominati nel seguito.

Merita ricordare che i programmi elencati qui sono presenti in qualunque distribuzione *completa* del sistema TEX. Bisogna però ricordare che la distribuzione MiKTEX di solito viene (imprudentemente) installata in forma ridotta e l'utente si affida troppo spesso alla capacità di questa distribuzione di auto-installarsi le parti mancanti; purtroppo alcuni utenti non attivano questa funzionalità, ma tutti al momento del bisogno possono trovarsi in una situazione in cui non hanno accesso alla rete, per cui queste auto-installazioni automatiche non possono avere luogo. Questa è una ragione in più per eseguire l'installazione completa di MiKTEX, perché contiene tutti questi programmi.

biber, BibTEX, bibtex8, bibtexu sono i programmi che permettono di gestire i database bibliografici (esistenti e predisposti a mano attraverso le funzionalità di alcuni *shell editor*, oppure con altri programmi specifici, come BibDesk e Jabref, non distribuiti con il sistema TeX) per estrarne i dati da riportare in una o più bibliografie, a seconda del documento e dello stile di

composizione richiesto; biber gestisce anche i dati codificati in UNICODE o UTF-8; BibTeX, come anche i suoi programmi affini, gestisce solo i dati codificati in una delle tante codifiche dove ogni carattere non facente parte del sotto-insieme ASCII a 7 bit viene codificato con 8 bit; per gestire la moltitudine di alfabeti basati su quello latino, che impiegano molti caratteri con diacritici sopra o sotto i segni di base, non bastano i 128 caratteri con l'ottavo bit posto a 1, e quindi esistono tante codifiche diverse a seconda dei caratteri che vi sono elencati.

makeindex e xindy sono programmi che permettono di gestire i dati raccolti dal programma di composizione per creare indici analitici, glossari, elenchi di acronimi e simili; il primo gestisce solo caratteri codificati a 8 bit; il secondo gestisce anche i caratteri UNICODE e permette di eseguire ordinamenti alfabetici anche per alfabeti complessi come quello del greco politonico. makeindex in realtà non ha nessun problema a elaborare file grezzi di indicizzazione prodotti da lualatex e xelatex mentre usano caratteri latini codificati in UNICODE.

dvips, dvipdfm, dvipdfmx, xdvipdfmx, ps4pdf, pstopdf, epstopdf, mptopdf servono per eseguire trasformazioni di formato per i file di uscita dei programmi di composizione, o per trasformare le immagini dal formato 'encapsulated PostScript' in formato 'portable document format' conservandone la natura vettoriale; oppure per trasformare i file prodotti da METAPOST, in file nel formato 'portable document format'.

METAFONT insieme a TeX, questo è il programma creato da Donald E. Knuth per produrre i font che il sistema TeX ha usato fin dai suoi albori; esso agisce su file di descrizione della collezione di segni del font scritti in un linguaggio particolare e salvati in file con estensione `.mf`; METAFONT elabora questi file e produce i file metrici necessari al programma di composizione per comporre il testo, nonché i file bitmapped che contengono in un formato particolare le matrici di pixel che formano il disegno di ogni segno; i file bitmapped servono per presentare il risultato della composizione sullo schermo oppure per stamparlo su carta; agli albori di TeX erano praticamente indispensabili; successivamente sono stati sviluppati i font Type 1, TrueType e OpenType e la loro necessità è virtualmente svanita. Tuttavia questo programma viene invocato automaticamente ogni volta che il programma di composizione o di trasformazione del formato non trova i file metrici oppure non trova i file bitmapped o quelli vettoriali necessari per presentare il documento in una forma leggibile sullo schermo o sulla carta. Talvolta l'esecuzione di questo programma viene richiesta a causa di un errore ortografico nei nomi dei font da usare oppure a causa di una installazione incompleta del sistema TeX; in questi casi METAFONT non riesce a portare a termine il suo compito; tuttavia, oltre a scrivere opportuni messaggi nel file `.log`, produce anche un file `missfont.log`, che l'utente trova nella stessa cartella del documento che sta componendo;

leggendo questo file l'utente può capire di quali font mancanti si tratti e può provvedere in merito.

METAPOST è un programma di disegno programmato che permette di eseguire disegni descritti mediante una forma semplificata del linguaggio PostScript; l'utente usa sostanzialmente lo stesso linguaggio di METAFONT, con poche importati ma ovvie modifiche dovute allo scopo dei disegni che produce, ma i comandi per disegnare sono sostanzialmente gli stessi. Uno dei vantaggi di METAPOST è che i suoi disegni, salvati in file con una estensione numerica, possono essere inglobati nei documenti da comporre qualunque sia il programma che si usa per comporli. Generalmente è necessario cambiare l'estensione numerica in `.mps`, oppure impostare METAPOST perché inserisca il numero progressivo non nell'estensione del file, ma nel nome stesso del file, così da usare costantemente l'estensione `.mps`.

pdfcrop serve per scontornare le immagini in formato 'portable document format' in modo che, inserite nel documento, non solo appaiano nel punto desiderato, ma non siano racchiuse fra spazi bianchi eccessivi.

arara è un programma in grado di leggere le prime righe scritte in modo speciale in un file sorgente da comporre con uno dei programmi di composizione del sistema TEX; esso è capace di eseguire una serie di funzioni in modo tale da comporre il documento finito indipendentemente dal numero di compilazioni necessarie e dall'esecuzione di programmi esterni a quello di composizione.

latexmk è uno script esterno che provvede a gestire la composizione completa di un documento; in un certo senso è simile ad arara, ma è più difficile da personalizzare; siccome esiste da molti anni, certa documentazione fa ancora riferimento a latexmk invece che ad arara, perciò è opportuno conoscerne l'esistenza in modo da poter configurare arara adeguatamente per svolgere le operazioni specifiche che sarebbe stato necessario personalizzare mediante latexmk.

fmtutil, fmtutil-sys sono i programmi per generare o rigenerare i file di formato; raramente può accadere di doversene servire, ma succede.

updmap, updmap-sys sono i programmi necessari per costruire gli elenchi dei font disponibili, sia bitmapped sia vettoriali, ciascuno con le sue particolarità, affinché i programmi di composizione o di trasformazione la cui uscita è in formato PostScript o in formato PDF conoscano la loro esistenza e li possano gestire ognuno secondo le sue particolarità.

mktexlsr serve per aggiornare i database dei nomi dei file facenti parte del sistema TEX; se questi database non sono aggiornati, il sistema TEX non riesce a trovare ed usare certi file importanti, cosicché i suoi programmi di composizione non possono completare il loro compito.

kpseaccess, kpsepath, kpsereadlink, kpsestat, kpsetool, kpsewhere, kpsewhich, kpsexpand sono i programmi specifici che il sistema TEX usa per accedere ai file elencati nei database dei nomi dei file generati con mktexlsr; anche l'utente li può usare dando gli appositi comandi nella finestra comandi (o prompt dei comandi); è molto utile sapersene servire e servirsene.

tex, etex, latex, pdftex, pdflatex, xetex, xelatex, luatex, lualatex sono alcuni programmi di composizione del sistema TEX di cui si è già parlato in questo capitolo; non sono gli unici; esiste anche arlatex per comporre in arabo, ptex per comporre in giapponese e nelle altre lingue orientali; mltex e mllatex servono per comporre documenti multilingue (questi programmi sono conservati per retrocompatibilità), aleph e lamed per comporre con font codificati UNICODE (anche questi conservati per retrocompatibilità; le loro funzioni sono state assorbite da xetex, xelatex, luatex, lualatex); context, e diversi altri programmi di composizione.

texdoc è il programma il cui nome viene usato come comando, da scrivere in una finestra comandi o prompt dei comandi, per leggere la documentazione dei pacchetti o dei programmi; tale documentazione è già installata mediante la distribuzione prescelta, quindi evita di dover andare a cercare in rete ciò che è già disponibile nel proprio disco dove è installato il sistema TEX.

asy è il nome del file eseguibile del programma Asymptote; si tratta di uno strumento per disegnare con grafica vettoriale, un po' come METAPOST, ma secondo alcuni molto migliore; consente anche di eseguire disegni tridimensionali, cosa che METAPOST non può fare. Viene spesso suggerito come una alternativa rispetto ad eseguire disegni con altri metodi, come per esempio *pgfplots* oppure *tikz*, che sono pacchetti di disegno programmato basati sul linguaggio LATEX.

Quasi tutti i programmi che fanno parte delle distribuzioni TEX sono documentati in inglese, alcuni, pochi, in polacco o in tedesco; la documentazione è leggibile con il programma texdoc. Quelli elencati sopra sono tutti documentati in inglese.

L'utente normale generalmente si serve di uno *shell editor* per elaborare i suoi file `.tex`; questi editor sono forniti di voci di menù a discesa o di icone da selezionare per eseguire molti dei programmi elencati sopra. Il processo è così trasparente per l'utente, che pensa che l'editor sia quello che fa tutto. Si raccomanda sempre di non confondere l'editor con i programmi che l'editor è capace di usare selezionando una voce da un menù o cliccando su una icona; queste funzionalità sono presenti nell'editor per la comodità dell'utente; alcune rare volte queste funzionalità preconfezionate non soddisfano le necessità dell'utente, che deve quindi essere in grado di aprire un terminale (finestra comandi o prompt dei comandi) e scrivere gli opportuni ordini in questa finestra. È quindi opportuno che l'utente conosca l'esistenza di questi programmi e, in caso di bisogno, sappia documentarsi e li possa usare per soddisfare le proprie necessità.

Capitolo 5

LaTeX: prime nozioni

5.1 Introduzione

Verranno ora esposte le prime nozioni per usare LaTeX e per comporre semplici documenti di solo testo. Sarà necessario mostrare esempi di codice sorgente, così come sarà necessario descrivere la 'sintassi' di alcuni comandi.

Il codice e i vari comandi saranno composti con il font della famiglia a spaziatura fissa, per esempio si scriverà `\documentclass`. La grammatica prevede che ogni comando possa avere degli argomenti il cui significato logico viene indicato da un nome o da una semplice locuzione scritta in corsivo e racchiusa fra parentesi 'acute', per esempio ⟨*classe*⟩, così che la sintassi del comando `\documentclass` possa essere espressa così:

`\documentclass[`⟨*opzioni*⟩`]{`⟨*classe*⟩`}`

Con questo tipo di grammatica gli argomenti che obbligatoriamente devono essere forniti ad un comando che ne richieda sono sempre racchiusi fra parentesi graffe, mentre gli argomenti facoltativi, opzionali, se sono presenti, sono sempre racchiusi fra parentesi quadre. Le parentesi acute della descrizione della grammatica servono solo per questa descrizione e non vanno effettivamente introdotte quando il comando viene usato.

5.2 L'inizio del file sorgente

Il comando `\documentclass` è il primo che deve essere dato all'inizio di un file (o nel primo file di un gruppo) da trattare con LaTeX.

La ⟨*classe*⟩ indica il tipo di documento che si intende comporre; le classi standard del sistema sono *`book`*, per comporre libri, *`report`* per comporre rapporti 'tecnici', e *`article`* per comporre articoli; esistono ovviamente altre classi non standard, ma queste tre sono le più note sotto ogni aspetto. Un conto

recente porterebbe ad un totale di circa 400 diverse classi fornite dal sistema TeX; è veramente difficile che un utente non riesca a trovare già preconfezionata una classe che porti alla composizione del suo documento come vuole lui, ma succede; si vedrà più avanti come personalizzare la propria composizione.

Per questo libro è stato usato il comando

```
\documentclass[b5paper]{book}
```

e con questa dichiarazione iniziale si è detto che si vuole comporre un libro e che lo si vuole stampare su carta ISO-UNI B5 (176 mm × 250 mm).

L'inizio del file sorgente prosegue facoltativamente con una serie di specificazioni per renderne più agevole la scrittura, per scegliere la lingua o le lingue da usare nel documento, per scegliere la codifica di default dei font da usare per comporre il documento, per estendere i comandi di default con istruzioni non standard al fine di ottenere risultati particolari.

Questo insieme di dichiarazioni preliminari si chiama *preambolo*; non è necessario che il preambolo contenga qualche cosa, ma in pratica il preambolo in Italia e per i documenti in italiano contiene almeno la specificazione dell'uso della lingua italiana e probabilmente la specifica dei font a 256 caratteri che contengono le lettere accentate; questo vale praticamente in ogni paese, tranne quando la lingua da usare è solamente l'inglese che per LaTeX è la lingua di default e la sua scrittura non contiene lettere accentate. Per questo libro la prima parte del preambolo comincia (in forma semplificata) così:

```
\documentclass[b5paper]{book}
\usepackage[utf8]{inputenc}%  codifica di ingresso
\usepackage[T1]{fontenc}%        codifica dei font
\usepackage[italian]{babel}% gestione delle lingue
\usepackage{guit}%                    logo del GuIT

\author{\GuIT}
\title{Introduzione all'arte\\
          della composizione tipografica con~\LaTeX}
\date{Versione 0.0}
```

Come vi si può leggere in chiaro, in questo preambolo si ordina a LaTeX di usare un certo numero di file di estensione mediante il comando `\usepackage`: in ordine gli si invoca il pacchetto per gestire l'immissione del testo, specificandogli come opzione che i caratteri immessi corrispondono alla codifica *utf8*; poi gli si ordina di usare il pacchetto per gestire la codifica (encoding) per i font di composizione del documento mediante la sigla *T1*, che corrisponde alla codifica standard di LaTeX per poter usare i caratteri latini accentati (vedi il capitolo 26); poi gli si comanda di usare il pacchetto *babel* per gestire lingue diverse dall'inglese e come opzione gli si specifica l'italiano; poi infine gli si ordina di usare il pacchetto *guit* per poter avere a disposizione il comando `\GuIT` per comporre la sigla del Gruppo degli Utilizzatori Italiani di TeX; volendo si potrebbe usare anche il logo

del gruppo. Si noti che il pacchetto *babel* per gestire le lingue è l'unico che si può usare quando la composizione viene eseguita con pdflatex; invece con xelatex e lualatex, pur non essendo vietato l'uso di *babel*, secondo chi scrive, è preferibile usare il file di estensione *polyglossia*, che mette a disposizione strumenti comodi per specificare la lingua principale e le lingue secondarie da usarsi all'interno dello specifico documento. In compenso per lavorare con xelatex e lualatex non si deve specificare la codifica di ingresso, e se si usano solo font OpenType, nemmeno la codifica dei font.

I primi tre pacchetti sono pressoché *obbligatori* per ogni documento composto con pdflatex in italiano e anche, specificando un diverso nome della lingua, per comporre documenti in qualunque lingua diversa dall'inglese; si possono specificare anche diverse lingue mediante una lista di nomi (in inglese) separati da virgole, l'ultima delle quali è quella usata di default nel documento.

Alcuni suggeriscono di usare la codifica *utf8* per tutte le macchine e ogni sistema operativo; alcuni *latin1*; altri per le macchine Windows suggeriscono la codifica *ansinew*; altri ancora suggeriscono la codifica *latin9*; per le macchine Macintosh 'vecchiotte' (con sistema operativo precedente al Mac OS X) alcuni preferiscono la codifica *applemac*. Questa questione della codifica verrà sviluppata più avanti nel capitolo 26; per ora si preferisca la codifica *utf8* che va bene per tutte le macchine ed è adatta per scrivere in qualsiasi lingua.

Per il file di uscita l'opzione *T1* è senz'altro consigliabile per scrivere in italiano, direi quasi obbligatoria. L'alternativa è di non specificare nulla; la differenza passerà inosservata al principiante, ma dopo un po' di esperienza anche il neofita si accorge che senza specificare nulla, usando quindi la codifica predefinita *OT1*, le lettere accentate del file di ingresso o non appaiono per nulla, oppure appaiono con gli accenti non perfettamente centrati sulla lettera di base, e che le parole contenenti lettere accentate tendono a produrre una giustificazione peggiore delle righe a causa del fatto che la divisione in sillabe è meno efficiente con la codifica *OT1*.

Dopo la riga vuota (o bianca) si specifica che l'autore di questo libro è il GuIT stesso, si specifica il titolo del libro e, invece della data, si specifica il numero della versione. 0.0 è il valore iniziale, ma è certo che il documento che state leggendo abbia un numero successivo, che non sia, cioè, la versione 'zero'.

Lavorando con xelatex o lualatex quelle quattro righe possono essere modificate così:

```
\documentclass[b5paper]{book}
\usepackage{polyglossia} % gestione delle lingue
\setmainlanguage{italian}%     lingua principale
\usepackage{fontenc}%          gestione dei font
\usepackage{guit}%                 logo del GuIT

\author{\GuIT}
\title{Introduzione all'arte\\
           della composizione tipografica con~\LaTeX}
```

```
\date{Versione 0.0}
```

Il preambolo di questo libro contiene altre cose che verranno descritte più avanti. Finito il preambolo comincia la composizione vera e propria del documento.

5.3 Il documento

Come è facile prevedere, il documento comincia con

```
\begin{document}
```

e, altrettanto prevedibilmente, finisce con

```
\end{document}
```

Tutto ciò che è racchiuso fra queste due righe viene composto o viene usato per controllare la composizione. È opportuno notare che tutto ciò che nel file compare dopo `\end{document}` non viene stampato ma può essere usato per scrivere dei commenti relativi al file, allo stato di avanzamento della composizione del documento, o ad altro che possa interessare il compositore per sua futura memoria.

In questo libro il documento comincia (nella versione 0.0 cominciava) così:

```
\begin{document}
\frontmatter
\maketitle
\tableofcontents

\chapter*{Presentazione}
```

dove il primo comando dichiara che si stanno componendo le pagine preliminari; il secondo comando serve per comporre la pagina del titolo usando le scarne informazioni circa l'autore, il titolo e la versione fornite nel preambolo; il terzo comando ordina di comporre l'indice generale; la riga bianca non conta niente, ma ha un effetto visivo di separazione nel file sorgente; in generale essa serve per marcare la fine di un capoverso, ma in questo punto non ci sono capoversi di nessun genere da comporre.

Il comando `\chapter*` serve per dare un titolo ad un capitolo non numerato e il cui titolo non viene inserito nell'indice generale né nelle testatine; nella parte iniziale anche il comando `\chapter` produce un capitolo non numerato, il cui titolo, però, compare nell'indice.

Come si vede il mark-up contiene un'unica piccola difficoltà per coloro che non hanno nessun rudimento di inglese, cioè che tutti i comandi sono in inglese; il buono è che, non contenendo abbreviazioni, questi comandi sono immediatamente comprensibili senza bisogno di ricorrere a elenchi di comandi criptici.

5.4 La fine del documento

Quando il testo del documento è stato completamente introdotto nel file sorgente, non resta che dichiararne la fine chiudendo l'ambiente *document* mediante il comando

```
\end{document}
```

Dopo questa riga e questa istruzione, si può scrivere quello che si vuole, ma non verrà mai composto; questa è la dichiarazione che il documento è terminato, non che il file è terminato.

5.5 Un semplice esercizio

Esercizio 5.1 Copiate dal testo precedente i comandi illustrati a partire dal comando `\documentclass`; dopo `\chapter*{...}` introducete un qualunque testo con capoversi sufficientemente lunghi, magari un capoverso in italiano, uno in inglese, uno in francese (non dovrebbe essere troppo difficile trovare dei libri o dei giornali che contengano delle frasi, anche in lingua straniera, da copiare, se la propria fantasia e la propria conoscenza delle lingue straniere non fosse sufficiente). Alla fine del documento ricordatevi di dichiararne la fine; sembra lapalissiano, ma è facile dimenticarsene.

Salvate il file scritto rigorosamente in formato testuale (meglio se facendo uso di uno *shell editor* adatto all'uso di LaTeX) per esempio con il nome `esempio1.tex` (il nome non è importante, ma l'estensione *deve* essere `.tex`.

Lanciate LaTeX; se state lavorando con uno shell editor[1], cliccate sull'opportuna icona; se non state lavorando con un shell editor aprite la finestra comandi o terminal o xterm o console, comunque si chiami sul vostro calcolatore quella finestra nella quale potete inserire i cosiddetti comandi in linea, scrivendo esplicitamente i nomi dei programmi da lanciare e del file su cui devono operare; in questo esempio scriverete

```
pdflatex esempio1
```

senza specificare l'estensione `.tex`.[2]

Se nella finestra comandi o nell'opportuna finestra dello shell editor compaiono messaggi d'errore, si riveda il testo perché gli unici errori che si possono aver fatto consisterebbero in errori di battitura; si riveda con particolare attenzione di aver usato le parentesi graffe dove ci vogliono le graffe, le parentesi quadre dove ci vogliono le quadre, e che tutte le

[1] D'ora in poi la locuzione inglese *shell editor* non sarà quasi mai scritta in corsivo, ma la si considererà acquisita come locuzione nomale per questo testo.

[2] L'estensione deve essere specificata in quei rari casi in cui si *debba* usare un'estensione diversa.

parentesi si accoppino correttamente, cioè che ad ogni parentesi aperta segua una parentesi chiusa dello stesso tipo, come in matematica; quindi: aperta–chiusa, aperta–chiusa,... oppure: aperta, aperta – chiusa, chiusa. Questo è uno degli errori più comuni; inoltre i comandi devono essere scritti correttamente: se ne riveda l'ortografia ricordando che il correttore ortografico dell'editor di solito non rileva e non corregge gli errori di battitura dei comandi.

Se le ultime righe di quel che compare nella finestra dicono che il file è stato correttamente salvato nel file `esempio1.pdf`[3] lanciate il previewer adatto a quel formato; potrà essere SumatraPDF se si dispone dell'installazione MiKTeX; potrà essere Okular se si sta operando su Linux, eccetera.

Ammirate la vostra composizione sullo schermo e osservate se il testo in inglese e/o in francese è stato composto bene, in particolare se le cesure in fin di riga sono corrette.

Questo semplice esercizio vi permette di rilevare quanto sia semplice usare LaTeX, specialmente se state usando uno *shell editor*; con quest'ultimo potete premere un solo pulsante che vi attiva l'esecuzione di LaTeX e, se non ci sono errori, vi attiva anche il *previewer*; ad ogni successiva compilazione generalmente la finestra del *previewer* viene automaticamente aggiornata, per cui avete una compilazione quasi sincrona; si consiglia di compilare sovente e di verificare sovente sul *previewer* che non ci siano errori grossolani di composizione, per esempio potreste aver dimenticato l'effetto di una dichiarazione e quindi potreste scoprire che da un certo punto in avanti state componendo con un font, perché vi siete scordati di ripristinare il font precedente. Questi sono errori grossolani che conviene correggere prima di avere composto le settecento pagine del libro che volete scrivere...

Gli errori più fini vanno ricercati nelle successive revisioni delle bozze, che, per fortuna, non richiedono di stampare su carta una serie di bozze dietro l'altra, ma possono essere riviste direttamente sullo schermo.

Va notato che molti *shell editor* permettono di eseguire sia la *forward search* sia la *reverse search*. Queste operazioni prevedono una interazione fra l'editor e il *previewer* in modo che azionando opportuni pulsanti dell'editor mentre il cursore si trova in una certa posizione nel testo del file sorgente, nella finestra del file compilato il cursore viene collocato se non nella stessa posizione, qualche parola più avanti o più indietro; questa è la *forward search*. Analogamente con il cursore nella finestra del *previewer* in una certa posizione, cliccando su un apposito pulsante o premendo una opportuna combinazione di tasti, il cursore si sposta nella finestra dell'editor praticamente nella stessa posizione, parola più, parola meno; questa è la *reverse search*. Grazie a questa interazione fra editor e *previewer* la correzione delle bozze procede spedita in quanto la finestra dell'editor e quella del *previewer* sono praticamente sincronizzate.

[3] Se state usando TeXShop su un Mac, il file salvato in formato PDF viene aperto automaticamente dal previewer.

È probabile che le parti di testo scritte in lingue diverse dall'italiano risultino avere alcune parole divise in sillabe in fin di riga (la cesura) in modo errato; questo dipende dal fatto che nel preambolo dell'esempio si è invocato il pacchetto per il trattamento delle lingue e gli si è specificato di voler comporre in italiano. È ovvio che la sillabazione delle altre lingue è diversa da quella dell'italiano, ed è altrettanto ovvio che scrivendo in lingue diverse dall'italiano è possibile che alcune cesure risultino errate.

Avete probabilmente usato, specialmente per il francese, molte lettere accentate; avrete quindi sfruttato a fondo le possibilità del vostro editor per inserire i caratteri accentati diversi da quelli che vi compaiono già sulla vostra tastiera, come per esempio 'ô' di 'diplôme'; molto probabilmente il file composto visualizzato nel *previewer* vi mostrerà il segno giusto.

5.6 I caratteri speciali

Per scrivere il mark-up secondo LaTeX bisogna ricordare che alcuni caratteri hanno un significato speciale; essi sono i seguenti[4]:

```
\  {  }  %  $  ^  _  &  #  ~
```

Il carattere \, chiamato *backslash* in inglese e *barra inversa* in italiano, serve come iniziatore di un comando o istruzione, come lo si preferisce chiamare; una istruzione è sempre costituita dal segno \ seguito senza spazi interposti da una *stringa di lettere minuscole e/o maiuscole* e questa stringa termina quando è seguita da un carattere non alfabetico; oppure una istruzione è formata dal carattere \ seguito da *un solo carattere non alfabetico*; oppure essa è costituita da un carattere *attivo*. Nella lista di caratteri speciali illustrata sopra il carattere ~ è un carattere attivo, ed è l'unico carattere attivo definito da LaTeX. Se ne possono avere altri, ma questo si può ottenere solo con file di estensione; per esempio con l'opzione per la lingua italiana di `babel`, o l'impostazione della lingua italiana per `polyglossia`, il carattere " è anch'esso attivo[5], come lo è per quasi tutte le lingue con la notevole eccezione dell'inglese.

I caratteri { e } sono molto particolari perché servono per delimitare gli argomenti dei comandi, ma svolgono anche altre funzioni di cui si parlerà in seguito.

Il carattere % svolge il compito di inizio di un commento: come in tutti i linguaggi di programmazione esiste una maniera per inserire dei commenti in linea con il codice, così anche con LaTeX l'inserzione del carattere % in una riga ordina al programma di composizione di trascurare tutti i caratteri che seguono

[4] In verità ci sarebbero anche i caratteri `‘, ’, "` che in certe circostanze svolgono funzioni speciali; il lettore non se ne preoccupi qui, ma sappia che in italiano il doppio apice svolge moltissime funzioni, tranne quello di doppie virgolette diritte.

[5] In verità sia con `babel` sia con `polyglossia`, il carattere " è preimpostato come un normale carattere analfabetico; per farlo funzionare come carattere attivo, bisogna attivarlo con gli appositi comandi dei due pacchetti per la gestione delle lingue.

`%` sulla stessa riga, fino al primo carattere diverso da uno spazio (noto anche con il nome di *blank*) nella riga successiva.

Il carattere `$` serve come 'interruttore' per ordinare a TeX di passare dal modo testo al modo matematico o viceversa. Quando l'interprete TeX è in modo testo interpreta i caratteri del file d'entrata in una certa maniera, mentre quando è in modo matematico, li interpreta come segni matematici. Per esempio, lo stesso carattere `-` in modo testo indica un trattino '-', mentre in modo matematico rappresenta il segno negativo '−', e si vede chiaramente che si tratta di due segni diversi; un analogo diverso modo di interpretare i caratteri del file di entrata viene usato da TeX per gestirli appropriatamente quando esso è in modo matematico.

Il segno `^` viene usato solo in matematica e indica che quanto segue eventualmente racchiuso fra parentesi graffe, rappresenta un esponente o un apice. Per usarlo come accento circonflesso nel testo basta farlo precedere da un backslash; per indicare un accento circonflesso in modo matematico bisogna usare il comando `\hat`.

Il segno `_` serve in matematica per indicare un deponente o un pedice. Per usarlo come 'underscore' sia nel testo sia in matematica basta farlo precedere dal backslash.

Il segno `&` serve per separare fra di loro i brevi testi o le brevi espressioni matematiche che entrano nelle celle di una tabella o di una matrice. Preceduto dal backslash serve per comporre l''e commerciale', *ampersand* in inglese.

Infine il segno `#` serve per marcare il numero progressivo dell'elenco dei parametri nella definizione di una nuova istruzione; preceduto dal backslash serve per comporre il *number sign* oppure *hash sign*, in italiano chiamato gergalmente 'cancelletto'.

Per scrivere questi caratteri in un brano di testo senza che svolgano il loro compito normale, bisogna farli precedere dal backslash. Ecco quindi che per esprimere un ammontare di denaro in dollari bisogna usare `\$` nel file sorgente per cui `\$234,99` diventa $234,99. Analogamente per indicare un incremento del 10% bisogna scrivere `10\%` nel file sorgente. L'unico carattere che non è stampabile ricorrendo al segno `\` è proprio questo stesso segno `\`. Per stamparlo bisogna ricorrere al comando `\textbackslash`. Siccome nel testo questo segno non si usa quasi mai, il fatto di dover ricorrere ad un comando così lungo non costituisce un vero problema. Generalmente sono disponibili anche i comandi `\textasciicircum` e `\textunderscore` che servono per inserire i rispettivi segni durante la composizione del testo. Esistono anche i comandi `\textdollar` e `\textasciitilde`, quest'ultimo per inserire una tilde invece di uno spazio non separabile, come avviene usando il carattere attivo `~`.

5.7 Testo strutturato

Questo paragrafo è preliminare a quello che segue, ma merita un suo spazio autonomo, perché strutturare un testo è una operazione che va fatta sempre,

anche con i documenti più semplici.

Persino una lettera commerciale è strutturata in campi disposti variamente su uno o più fogli; lo spazio per il nome e l'indirizzo del destinatario, quello per il nome e l'indirizzo del mittente, i riferimenti d'archivio, l'oggetto della lettera, l'apertura, il testo, la chiusura, l'eventuale elenco degli allegati, l'elenco dei destinatari in copia, eccetera. Ognuna di queste strutture può venire posta in riquadri o comunque in posizioni adeguatamente scelte dallo stile di scrittura preferito. Spesso le indicazioni del mittente sono prestampate sul foglio mediante l'uso di *carta intestata*; l'intestazione può contenere anche il logo dell'azienda e altre informazioni non elencate sopra.

Certo, una lettera ha la sua struttura e non è divisa in capitoli, paragrafi, sottoparagrafi, e simili. Altri documenti, invece, sono suddivisi con questi sezionamenti; altri ancora, come un dizionario, una Bibbia, un Codice Penale, sono fortemente strutturati, ma diversamente da come descritto per un comune libro.

Usando il sezionamento prodotto da capitoli, paragrafi, sottoparagrafi, bisogna stare attenti ad alcune regole generali che sarebbe bene osservare sempre, anche se talvolta sembra necessario fare delle eccezioni.

Ogni sezione dovrebbe contenere almeno due sezioni di rango inferiore perché queste ultime abbiano un senso; quindi non si divide in parti un documento che contenga un solo capitolo, tenendo conto che la divisione in parti è sempre facoltativa. Ma non si divide in paragrafi un capitolo che contenga un solo paragrafo, e via di questo passo. Perciò per un capitolo la divisione in paragrafi richiede sempre almeno due paragrafi; per ogni paragrafo sono sempre richiesti almeno due sottoparagrafi; eccetera. Nessuno vieta, naturalmente, di cominciare un capitolo con alcuni capoversi non preceduti da un sezionamento, ma aventi carattere introduttivo, per poi cominciare il sezionamento con i paragrafi. Lo stesso vale per ogni livello di sezionamento.

Ma come ogni sezione deve contenere almeno due sezioni di rango inferiore, deve anche essere chiaro quando la sezione termina e ne comincia un'altra; non ha quindi senso cominciare un paragrafo e dopo alcuni capoversi aprire un sottoparagrafo; chiuso questo, riprendere li discorso del paragrafo, con alcuni capoversi, e poi aprire un altro sottoparagrafo. Procedere in questo modo fa perdere al lettore il senso della strutturazione. Perciò una volta cominciata la strutturazione con sezioni di rango inferiore, la si termina con l'apertura di una sezione di rango relativo superiore completa con il suo numero (se è richiesta la numerazione) e il suo titolo.

Inutile dire che non si possono saltare livelli di sezionamento; aperto il capitolo si procede con i paragrafi[6]; dentro i paragrafi si procede con i sottoparagrafi; dentro i sottoparagrafi, se davvero fosse necessario scendere a livelli così bassi, si procede con i sotto sottoparagrafi.

Quindi prima di sezionare, conviene programmare bene come segmentare il testo per non trovarsi poi con situazioni difficili da gestire al fine di mantenere

[6]In questo paragrafo si è usata la parola *paragrafo* piuttosto spesso. Il lettore, naturalmente non avrà confuso il paragrafo con il *paragraph*...

la coerenza della strutturazione.

5.8 Organizzazione dei file sorgente

Questo paragrafo andrebbe scritto in fondo a questo testo, ma tutto sommato è logicamente connesso con il materiale esposto in questo capitolo. All'occorrenza potete saltarne la lettura, salvo poi tornare a questo paragrafo quando dovrete gestire la composizione di un lungo testo molto strutturato.

I semplici esercizi che si possono fare ora e con il contenuto dei primi capitoli che seguono, comportano la scrittura di brevi file sorgente che raramente, una volta compilati, superano il paio di pagine di testo composto.

Il file sorgente non richiede, quindi, una particolare organizzazione, perché la sua suddivisione naturale in 'preambolo' e 'corpo del documento' è più che sufficiente.

Tuttavia si immagini di dover scrivere un testo diviso in numerosi capitoli, appendici, indici e glossari, prefazioni e postfazioni, eccetera. Il file sorgente monolitico rischierebbe di diventare enorme, senza contare l'organizzazione dei file da includere, come i file che contengono le immagini, o i file che conviene predisporre a parte perché richiedono una paziente messa a punto del contenuto.

In questo caso l'organizzazione monolitica non è consigliabile, e infatti LaTeX offre diversi strumenti per spezzare il file sorgente in file di minori dimensioni. LaTeX, infatti, offre due insiemi di comandi per questo scopo: `\input` e `\include` assieme a `\includeonly`, oltre al comando `\includegraphics` per immettere nel flusso di informazioni da elaborare i file contenenti le immagini. Per quest'ultimo comando si veda più avanti il capitolo 10. Per gli altri comandi la sintassi è la seguente:

```
\input{⟨file⟩}

\includeonly{⟨lista di file⟩}
\include{⟨file⟩}
```

L'informazione ⟨*file*⟩ è solamente il nome di un singolo file, mentre ⟨*lista di file*⟩ è un elenco di nomi di file separati da virgole. Per tutti i nomi dei file l'estensione `.tex` è sottintesa e quindi questa deve essere specificata se si vuole inserire un qualunque altro file con una estensione diversa, compresa l'estensione `.ltx` che in teoria dovrebbe essere l'estensione di default per tutti i file scritti con il mark-up LaTeX.

Merita invece osservare che quasi tutti i sistemi operativi ammettono che i nomi dei file possano contenere anche spazi e altri punti oltre a quello che separa l'estensione. Un nome di file del tipo `parte II.tex` oppure `parte.2.tex` è perfettamente lecito con moltissimi sistemi operativi. Ma è altrettanto ovvio che gli spazi e i punti, come anche gli altri caratteri che in TeX hanno un significato particolare, ma sono leciti nei nomi dei file, possono produrre guai

seri. Si sconsiglia, per esempio, di usare la linea ribassata (*underscore* in inglese), "_", perché per TEX quel segno indica i pedici in matematica ed è illegale in modo testo. Oggi spesso questo segno viene gestito correttamente se è presente in un nome di file, tuttavia se ne sconsiglia l'uso, e se si desidera avere un "allontanamento" visivo delle parti del nome del file si preferisca il trattino normale `-`. Con la versione del 2020 del linguaggio LaTeX (vedi il capitolo 30) Ora i programmi del sistema TEX accettano anche nomi contenenti spazi; tuttavia lo si sconsiglia. Come detto sopra, esistono altri modi per separare le 'parole' che formano il nome di un file.

Vale lo stesso discorso per il punto che separa l'estensione di una file dal suo nome, il quale potrebbe contenere un altro punto; evitando queste doppie interpretazioni dello stesso carattere si evitano molti problemi Perciò si evitino assolutamente i punti; se ci si riferisce al file `parte.2.tex` il ⟨*file*⟩ potrebbe essere `parte.2` ma TEX potrebbe scambiare `.2` per l'estensione e lamentarsi che non trova il file.

Si potrebbe racchiudere il nome del file fra doppi apici: `"parte II"` oppure `"parte.2"` ma c'è il problema che il doppio apice in quasi tutte le lingue invocate con *babel*, italiano compreso, è definito come carattere attivo, cioè è definito in modo che si comporti come una istruzione, non come un segno grafico; per questo motivo nel momento in cui si vuole leggere un file esterno mediante `\input` o `\include`, l'istruzione corrispondente al doppio apice viene eseguita e TEX si ritrova confuso e non sa che cosa leggere cosicché emette dei messaggi d'errore che talvolta sono difficili da associare al file che TEX non riesce a leggere.

Ci si ricordi infine che in molti sistemi operativi le lettere maiuscole e minuscole sono "diverse" al fine di identificare e distinguere i file. Per i sistemi UNIX e Linux i file `PartePrima.tex` e `parteprima.tex` sono due file diversi; nei sistemi Windows quei due nomi identificano lo stesso file.

Per evitare gli inconvenienti menzionati sopra si consiglia di usare nomi di file formati da una sola parola alfanumerica, senza punti e spazi inclusi, seguita dalla eventuale estensione, possibilmente scritti completamente con lettere minuscole. In questo modo non bisogna ricorrere a nessun espediente per gestire questi nomi durante l'esecuzione di una compilazione, ma, ancora più importante, non si generano inconvenienti quando si scambiano i propri file con collaboratori che lavorano con altre macchine e/o con altri sistemi operativi. La *portabilità* è un bene prezioso che gli utenti del sistema TEX apprezzano moltissimo.

Il comando `\input` ordina a LATEX di leggere il ⟨*file*⟩ specificato; il flusso di dati che viene letto, viene inserito nel flusso complessivo del documento come se fosse inserito esattamente al posto di `\input`; che questo ⟨*file*⟩ contenga una sola parola, o un intero paragrafo o capitolo non ha nessuna importanza; il suo contenuto viene letto incondizionatamente ed elaborato immediatamente.

Questo comportamento del comando `\input` si mantiene anche se è contenuto dentro altri file a loro volta inseriti nel flusso di elaborazione dati mediante altri comandi `\input` o anche comandi `\include`; non esiste un limite teorico al numero di file che possono essere inclusi richiamandosi in catena, salvo la limitazione ovvia delle dimensioni finite della macchina di elaborazione e del

numero di canali di ingresso che possono essere aperti contemporaneamente; questo numero massimo di canali è di sedici unità, ma alcuni di questi canali sono già impegnati per altre funzioni di servizio, per cui si può contare su una decina di canali effettivamente disponibili; questo numero dovrebbe essere sovrabbondante rispetto a qualunque esigenza insolita di immissioni a catena mediante comandi `\input`.

Il gioco di `\includeonly` e `\include` è invece più complesso; innanzi tutto, in assenza di `\includeonly` e della sua ⟨*lista di file*⟩, `\include` agisce quasi come `\input`; l'unica differenza è che prima di elaborare il contenuto del file incluso, LaTeX ordina di eseguire un salto di pagina, cioè di iniziare una nuova pagina. Questo permette di capire subito che `\include` non è usabile al posto di `\input`, proprio a causa di questo salto di pagina. Il comando `\include` è più adatto per includere un intero capitolo (che comincia sempre su una pagina nuova) piuttosto che per immettere semplicemente del testo da elaborare nel punto in cui compare il comando.

Ma tutto diventa più interessante e utile se si fanno agire `\includeonly` ed `\include` assieme; infatti in questo modo `\include` include solo i ⟨*file*⟩ che compaiono nella ⟨*lista di file*⟩; però, se l'argomento di `\include` è un ⟨*file*⟩ che non compare nella ⟨*lista di file*⟩, `\include` ordina a LaTeX di leggerne solo il file `.aux`, cioè il file ⟨*file*⟩`.aux`, che contiene le informazioni relative ai riferimenti incrociati e al contenuto dei vari contatori che interessano LaTeX, cosicché anche se il file ⟨*file*⟩`.tex` non viene elaborato, le sue informazioni utili sono tenute in conto e tutto procede agli effetti esterni come se il file fosse stato elaborato. Va da sé che se questo file ⟨*file*⟩`.tex` non è mai stato elaborato prima, non esiste nemmeno il file ⟨*file*⟩`.aux`, ma `\include` non protesta perché si limita a scrivere un avvertimento nel file `.log` privo di qualunque conseguenza sul buon fine della compilazione.

Un file incluso mediante `\include` può contenere a sua volta comandi `\input`, i file dei quali a loro volta... Va da sé però che, se un file non viene incluso perché il suo nome non compare nella ⟨*lista di file*⟩, non vengono elaborati nemmeno gli eventuali file subalterni richiamati dai comandi `\input`.

Al contrario di `\input`, i comandi `\include` non possono essere annidati l'uno dentro l'altro ed è del tutto logico che sia così.

Tutto questo può sembrare oltremodo complicato, mentre un esempio permette di rendersi conto che non solo è semplicissimo, ma è anche molto logico. Si pensi ad un lungo saggio che si ritiene di suddividere nei file seguenti:

1. `premesse.tex` che contiene le informazioni per comporre tutto il materiale iniziale, dall'occhiello fino alla prefazione, inclusi tutti gli indici che si vogliono creare.

2. `introduzione.tex` che contiene un capitolo introduttivo.

3. `statodellarte.tex` che contiene una dettagliata descrizione della situazione pertinente prima della teoria esposta nel saggio.

4. `motivazioni.tex` dove si descrivono le motivazioni che portano allo sviluppo della nuova teoria descritta nel saggio.
5. `sviluppo.tex` dove si discute lo sviluppo della nuova teoria.
6. `risultati.tex` dove si discutono i risultati alla luce della nuova teoria.
7. `confronti.tex` dove si discutono e si confrontano i nuovi risultati con quelli che si potevano ottenere applicando le teorie precedenti. Magari questo file contiene anche le considerazioni conclusive.
8. `appendici.tex` dove si raccolgono tutte le informazioni, dalla bibliografia alle informazioni accessorie allo sviluppo della nuova teoria, fino, eventualmente, agli indici analitici e ai glossari.

Si procederà allora a predisporre un 'master file' `nuovateoria.tex` nel quale si mettono solo il preambolo e i comandi per l'inclusione dei vari file:

```
% file nuovateoria.tex
%
\documentclass[⟨opzioni⟩]{⟨classe⟩}
⟨preambolo che contiene tutti i comandi per i pacchetti da usare, le definizioni di
nuovi comandi, e si conclude con:⟩
%
\includeonly{%
premesse,%
introduzione,%
statodellarte,%
motivazioni,%
sviluppo,%
risultati,%
confronti,%
appendici%
}
%
\begin{document}% Qui finisce il preambolo e comincia il documento

\frontmatter
\include{premesse}
\mainmatter
\include{introduzione}
\include{statodellarte}
\include{motivazioni}
\include{sviluppo}
\include{risultati}
\include{confronti}
\backmatter
\include{appendici}
```

```
\end{document}
```

Se si elabora il file `nuovateoria.tex` così come è presentato sopra, LaTeX legge, se ci sono, tutti i file dichiarati come argomento ai vari comandi `\include`. Ma si osservi come è scritta la ⟨*lista di file*⟩ che costituisce l'argomento di `\includeonly`; ogni file della lista è scritto all'inizio di una nuova riga la quale è terminata dal segno di commento al fine di non lasciare spazi spuri nella lista. In questo modo c'è la possibilità di inserire un segno di commento all'inizio delle righe che contengono i nomi dei file che *non* si desidera elaborare.

All'inizio della scrittura dell'intero documento si comincerà presumibilmente con il file `premesse.tex`; si metterà allora un segno di commento davanti al nome di tutti gli altri file; quando poi si passa a scrivere `introduzione.tex`, si toglie il segno di commento davanti al nome di questo file e lo si inserisce, invece, davanti al nome di `premesse.tex`. Via via che si procede, si compone un file alla volta e se ne controlla il risultato attraverso il programma di visualizzazione adeguato al tipo del file di uscita. Ogni volta la compilazione ha luogo su un solo file ed è particolarmente veloce; nello stesso tempo i riferimenti incrociati a elementi contenuti nei capitoli precedenti vengono tutti risolti correttamente. Anche le bozze, se le si volesse stampare, hanno le pagine numerate correttamente. Solo alla fine, a lavoro quasi terminato, si tolgono tutti i segni di commento davanti ai nomi dei file della ⟨*lista di file*⟩, e si esegue la composizione dell'intero testo in un unico file di uscita.

Se lo shell editor lo richiede (e quasi tutti gli shell editor lo richiedono) bisogna solo specificare all'editor il nome del 'master file' affinché, quando si compila e l'editor ha aperto solo uno dei file da includere, esso possa lanciare LaTeX sul 'master file', e non sul file correntemente aperto.

Con gli shell editor che comprendono i commenti di autoconfigurazione (le righe magiche) non è necessario fare nulla di speciale perché le specifiche righe magiche contenute in testa ad ogni file incluso dicono esplicitamente all'editor come si chiama il 'master file' su cui eseguire la compilazione.

Questo tipo di organizzazione del file d'entrata diventa più comodo da usare via via che si acquisisce pratica e se ne apprezzano i vantaggi.

Vale la pena di sottolineare che la divisione in capitoli da includere con il comando `\include` va benissimo con i capitoli. Il comando `\input` serve per gli scopi indicati, ma l'esperienza insegna a servirsene con il massimo profitto in diverse altre circostanze; per esempio lo scrivente ha sviluppato l'abitudine di comporre grandi tabelle in file a se stanti da includere con `\input`. Questo modo di procedere deriva dalla constatazione che le grandi tabelle sono complesse da comporre ed è facile commettere errori. Inserite in file a se stanti, e creato un unico piccolo 'master file' che ordini la lettura solo del file contenente la tabella, ogni tabella viene composta e corretta per suo conto e non c'è pericolo che gravi errori, particolarmente difficili da individuare, fermino la compilazione di un grosso file in un punto che non ha nulla a che vedere con la posizione dell'errore, come, ahimè, talvolta succede.

Dalla versione 0.09 di questa guida anche i file che compongono questo testo sono gestiti con il meccanismo di `\include` e `\includeonly`. Quest'ultimo,

collocato nel preambolo, è formato così:

```
\includeonly{%
 preliminari,%
 presentazione,%
%
 sincrona-asincrona,%
 nozioni-tipografia,%
 ortografia,%
 installazione,%
 prime-nozioni,%
 classi,%
 testi-speciali,%
 tabelle,%
 figure,%
 importazione-figure,%
 bibtex,%
 indici-glossari,%
 presentazioni,%
%
 matematica1,%
 matematica2,%
 microgiustificazione,%
 nozioni-tipografia,%
 ortografia,%
 caratteri,%
 classi,%
 nuovi-comandi,%
 layout,%
 documentazione,%
%
 pdfarchiviabile,%
 conversione,%
 simbologia,%
 divisione-sillabe,%
 codifiche,%
 PageOutput,%
 errori,%
 sintassi-riepilogo,%
 aggiornamenti-latex,%
 bibliografia,%
 indanalitico%
}
```

e, come si vede, ogni capitolo, numerato o non numerato, e ogni appendice viene elencata nella ⟨*lista di file*⟩ nella speciale maniera descritta sopra. Successivamente

il corpo del testo è formato dal blocco:

```
\begin{document}
%
\frontmatter
%
\include{preliminari}
\include{presentazione}
%
\mainmatter
\pagestyle{headings}
%
\include{sincrona-asincrona}
\include{nozioni-tipografia}
\include{ortografia}
\include{installazione}
\include{prime-nozioni}
\include{classi}
\include{testi-speciali}
\include{tabelle}
\include{figure}
\include{importazione-figure}
\include{bibtex}
\include{indici-glossari}
\include{presentazioni}
%
\include{matematica1}
\include{matematica2}
\include{filologia}
\include{presentazioni}
\include{microgiustificazione}
\include{caratteri}
\include{nuovi-comandi}
\include{layout}
\include{documentazione}
%
\include{pdfarchiviabile}
\include{conversione}
\include{simbologia}
\include{divisione-sillabe}
\include{codifiche}
\include{PageOutput}
\include{errori}
\include{sintassi-riepilogo}
\inlude{aggiornamenti-latex}
```

```
\cleardoublepage
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\backmatter

\include{bibliografia}
\include{indanalitico}
%
\end{document}
```

I comandi `\frontmatter` e `\mainmatter` specificano l'inizio dei preliminari e l'inizio del corpo del testo e sono presenti anche se gli eventuali file che dovrebbero venire inclusi di fatto non lo sono perché eliminati con segni di commento dalla lista dei file di `\includeonly`. Nelle prime versioni questa guida conteneva numerose appendici e al posto del comando `\backmatter` si era usato il comando `\appendix` che consentiva la numerazione delle appendici con lettere maiuscole. Successivamente le appendici sono diventate capitoli normali e nella `\backmatter` sono rimasti solo la bibliografia e i sei indici analitici. Inoltre questa impostazione rende più facile la separazione in tre tomi, come viene fatto dal 2013.

5.9 Gestione degli errori

Purtroppo talvolta i file sorgente contengono errori; in queste circostanze il programma di compilazione procede ad interpretare il flusso di informazioni che via via legge dai file sorgente, finché riesce a dare loro un qualche significato, poi si ferma con un messaggio d'errore. Quasi sempre si tratta di un comando scritto con un errore di ortografia; sullo schermo appare allora il messaggio d'errore con una qualche frase esplicativa e la linea dove il programma si è bloccato scritta 'spezzata' in corrispondenza dell'ultimo oggetto che esso ha letto. Se si tratta di un errore di ortografia in un comando, appare appunto il comando errato alla fine del primo moncone della riga presentata e la riga del flusso d'entrata non ancora sottoposto al processo di interpretazione appare subito sotto. Il primo moncone di riga è anche preceduto dal numero della riga del file sorgente.

Bisogna osservare che questi arresti ad ogni errore si manifestano solo se lo shell editor ha imposto al compilatore di comporre in 'error stop mode'; TeXShop, per esempio è configurato in questo modo; TeXstudio è invece configurato per comporre in 'non stop mode' (vedi poco più avanti).

Le risposte possibili ad un messaggio d'errore sono quelle di digitare un dei seguenti Tasti: Invio, x, q, s, h, i, e, eventualmente seguito da un stringa di testo.

Invio Premendo il tasto Invio il compilatore ignora l'errore e prosegue la compilazione fino all'errore successivo o fino alla fine del documento, se non incontra altri errori gravissimi. Di solito non è conveniente premere

il tasto Invio, ma è preferibile premere i tasti corrispondenti alle altre funzionalità del compilatore.

x Serve per terminare subito la compilazione; successivamente si può dare il comando allo shell editor di collocare il cursore nella riga contenente l'errore in modo da poterlo correggere e riprendere la compilazione dall'inizio.

q Serve per ordinare al compilatore di continuare in modo 'quieto', cioè senza fermarsi ad ogni errore e senza emettere messaggi d'errore. Ovviamente se a un certo punto il compilatore incontra un errore grave che lo blocca permanentemente, esso si ferma presentando come segnale di prompt un asterisco, non più un punto interrogativo; l'asterisco richiede che si immetta qualche cosa che il compilatore riesca a interpretare; non facile da dirsi, anzi questa è una condizione d'errore gravissimo che non ammette recuperi. Molti shell editor hanno un tasto o un bottone da cliccare che consente di 'abortire' la compilazione in modo pulito.

s Serve per ordinare al compilatore di proseguire la compilazione in 'scroll mode', vale a dire proseguendo la compilazione come in 'quiet mode', ma scrivendo sulla console i messaggi d'errore; se si ferma con un asterisco bisogna procedere come descritto sopra.

h Serve per chiedere aiuto; i messaggi informativi che il compilatore fornisce sullo schermo talvolta permettono di capire di che errore si tratti; talvolta sono troppo generici e bisogna armarsi di pazienza e di ragionamento investigativo per trovare che cosa c'è che non va.

e Serve per uscire dal programma di compilazione e di ritornare immediatamente allo shell editor con il cursore già messo nella riga con l'errore; questa operazione ha esito felice se c'è una perfetta integrazione fra compilatore e shell editor; ma talvolta non riesce perché lo shell editor non è configurato correttamente.

i Serve per introdurre una stringa di testo *al posto* dell'ultimo oggetto letto dall'interprete; per esempio se si scrivesse `\iput`, il compilatore si fermerebbe dopo aver letto questo oggetto a cui non sa dare un significato, perché è un comando scritto male; al prompt (costituito da un punto interrogativo) basta scrivere `i\input` e poi premere il tasto di invio; il compilatore sostituisce `\input` all'oggetto ignoto `\iput` e procede nella compilazione; si noti che il file sorgente non è stato toccato da questa operazione, quindi l'errore esiste ancora e bisogna correggerlo; bisogna cioè ritornare allo shell editor e chiedergli di collocarsi alla riga che contiene l'errore.

Se si ha l'accortezza di lanciare il compilatore abbastanza sovente, ogni possibile errore, anche difficilissimo da trovare, può trovarsi solo nel testo sorgente introdotto fra la penultima e l'ultima volta che si è eseguita la compilazione; se questa viene eseguita sovente, il testo introdotto è relativamente corto e quindi risulta più facile trovare l'errore.

Per errori più complessi bisogna ricorrere a pacchetti oppure a comandi

specifici, la cui discussione è fuori luogo in questo capitolo introduttivo destinato alle prime nozioni; per comprendere meglio certi errori e certi avvisi di cattiva composizione (`Overfull hbox...` e altri avvisi simili), conviene leggere il capitolo 27. Il trattamento degli errori più gravi e gli strumenti disponibili con il sistema TEX sono trattati nel capitolo 28.

Capitolo 6

LaTeX: i vari tipi di documenti e stili di composizione

6.1 Introduzione

I file di classe, che servono per definire i parametri compositivi del particolare documento che si intende comporre, sono il cuore di LaTeX e sono proprio loro che interpretano le istruzioni LaTeX per eseguire la composizione in accordo con lo stile desiderato.

6.2 Classi standard

LaTeX viene distribuito con una piccola dotazione di file di classe abbastanza generici da poter affrontare la composizione di qualunque documento; tuttavia questa genericità ha favorito lo sviluppo di classi particolari di cui si dirà brevemente nei prossimi paragrafi.

`book` è la classe con cui è stato composto questo testo. Essa prevede di usare di default tutte le strutture di sezionamento, da `\part` fino alla più minuta suddivisione del testo costituita da `\subparagraph`; prevede la composizione sul recto e sul verso dei fogli, prevede la pagina contenente il titolo come pagina a sé stante; il disegno grafico prevede uno spostamento della gabbia verso il margine interno; le testatine contengono il titolo corrente e il numero della pagina; il titolo corrente è scritto in caratteri maiuscoli inclinati; il documento può essere suddiviso in materiale iniziale, corpo del testo e materiale finale, con caratteristiche compositive adeguatamente diverse, eccetera.

report è la classe con cui si scrivono i rapporti 'tecnici': testi relativamente brevi, di solito non superiori ad una cinquantina di pagine, con il titolo facente parte della prima pagina dove comincia anche il testo; di default sono composti sul recto delle pagine, per cui la gabbia del testo è centrata. Non prevede la definizione dei comandi `\frontmatter`, `\mainmatter` e `\backmatter` perché un rapporto non è così strutturato come un libro, ma per il resto la classe assomiglia abbastanza alla classe *book*.

article è la classe con cui vengono composti brevi articoli, di solito di lunghezza non superiore alla decina di pagine, con il titolo sulla stessa pagina dove comincia il testo; spesso sono scritti su due colonne; hanno esplicitamente l'ambiente *abstract* per redigere un breve sunto del contenuto; non è definito il comando `\chapter` anche se continua ad essere definito il comando `\part`; nessuno vieta di dividere un articolo in parti, ma la struttura di sezionamento più alta è `\section`; spesso gli articoli sono senza indice generale, né indice analitico, ma dispongono quasi sempre di una bibliografia.

letter serve per comporre lettere commerciali; lo stile è molto americano, ma non è così difficile personalizzarlo per conformarsi allo stile italiano delle lettere commerciali; consente, volendo, di scrivere lettere con il medesimo testo ad una moltitudine di destinatari, così come consente di scrivere diverse lettere a destinatari diversi inserendole nello stesso file sorgente.

ltnews serve per comporre una semplice newsletter, talvolta di una sola pagina, composta su due colonne; vedi le *latex news* di cui si parlerà nel capitolo 30

ltxdoc serve per comporre la documentazione dei file sorgente dei pacchetti che fanno parte delle distribuzioni TEX Live e MiKTEX, o che possono venire distribuiti isolatamente da utente a utente; questi file sorgente contengono simultaneamente la documentazione e il codice e possono servire per produrre diverse classi e/o file di estensione; il loro formato è molto particolare, ma sono preziosi per i programmatori per poter scrivere dei programmi in linguaggio TEX o LATEX in una forma ben documentata, senza che questi commenti disturbino il lavoro del motore di composizione quando questo deve comporre un documento. Siccome da un file `.dtx` si possono estrarre un certo numero di file correlati, che quindi sono 'impacchettati' assieme dentro tali file, è consuetudine chiamare *pacchetti* i *file di estensione*.

ltxguide serve per comporre le guide di documentazione dei vari aspetti di LATEX; si tratta delle guide incluse in ogni distribuzione e si trovano generalmente nella cartella `.../doc/latex/base/`; queste guide sono la fonte principale di informazione che l'utente dovrebbe consultare e 'conoscere a memoria'...

`minimal` è una classe minima, fatta apposta per avere una classe che garantisca il minimo di performance, essenzialmente pensata per collaudare la definizione di nuovi comandi o lo sviluppo e le prestazioni di nuovi pacchetti di estensione quando il programmatore li sta costruendo.

`proc` è una classe simile ad `article` che serve per produrre degli articoli secondo uno stile da *proceedings* (atti di una conferenza).

`slides` serve per predisporre dei lucidi di presentazione di una lezione o di una conferenza. In realtà una volta si usavano le lavagne luminose; oggi si usano i videoproiettori; con questi ultimi non si producono lucidi ma si proietta direttamente dal calcolatore; le prestazioni di `slides` sono adatte proprio ai lucidi su fogli di metacrilato. Per le presentazioni oggi si possono usare diversi programmi, ma, restando in ambito LaTeX, esistono delle classi, come `beamer`, ottenuta con il pacchetto `beamer`, che producono delle presentazioni a colori bellissime, con animazioni finalizzate alla presentazione, e con la qualità della composizione della matematica e della gestione/importazione di immagini efficaci come quelle che solo LaTeX può offrire.

6.3 La creazione di nuove classi

L'utente può crearsi nuove classi con due approcci (in verità non mutuamente esclusivi) molto semplici; (*a*) modificare le prestazioni di una classe esistente e (*b*) creare una classe ex novo.

Il primo approccio è decisamente il più semplice ed è affrontabile da chiunque; il file `clsguide` che si trova nella solita cartella `.../doc/latex/base/` contiene le indicazioni per seguire questa via e presenta non pochi esempi.

Il secondo approccio è decisamente molto più complesso, ma consente una personalizzazione maggiore. A questo proposito può essere interessante leggere la documentazione relativa alla classe `octavo` in `.../doc/latex/octavo/octavo.pdf`. Questa classe è un rifacimento della classe `book` dove però sono state fatte scelte particolari sia nel dimensionamento della gabbia e delle altre informazioni 'geometriche' per la composizione, ma sono stati anche ridefiniti numerosi comandi per ottenere risultati particolari; usando la classe `octavo` al posto della classe `book`, il compositore non deve imparare nessun comando nuovo; al massimo deve familiarizzarsi con le nuove opzioni per il formato della carta che si riferiscono alla terminologia tradizionale britannica. Se nel preambolo di un documento da comporre con la classe `book` si sostituisce la parola `book` con la parola `octavo`, la composizione risulta assai simile, salvo la giustezza e la densità delle righe che risultano perfettamente adattate al corpo dei caratteri usati; ma se si specifica qualche opzione scelta fra quelle nuove, l'aspetto della composizione cambia in modo abbastanza significativo. La documentazione è molto istruttiva e spiega sia i motivi delle scelte fatte sia come queste siano state rese nel linguaggio di programmazione LaTeX; il compositore che vuole

cimentarsi nella creazione di nuove classi ha molto da imparare dalla lettura di questa documentazione.

Una via di mezzo è quella di utilizzare classi non standard ma fortemente personalizzabili come si vedrà nel prossimo paragrafo.

Infine, la via maestra è quella di non creare nessuna classe nuova, a meno che non sia fortemente indispensabile. Esistono infiniti pacchetti di estensione per modificare le prestazioni delle classi esistenti; qui se ne citeranno solamente alcuni, quelli più usati e considerati più utili.

fancyhdr serve per personalizzare le testatine ed i piedini; di ognuno si può personalizzare quello che compare nella pagina di sinistra, oppure nella pagina di destra; di ognuno si può specificare che cosa scrivere nel centro, a sinistra, oppure a destra, così da avere sei posizioni in ogni pagina per inserire informazioni che, in definitiva, aiutano a navigare nel testo, e quindi sono particolarmente indicate nei documenti molto strutturati. Mentre le testatine 'ordinarie', per esempio della classe ***book***, oltre al numero della pagina, possono contenere solo il titolo (corto) del capitolo o del paragrafo correnti, con ***fancyhdr*** si riesce anche a inserire informazioni di terzo livello, per esempio il titolo dei sottoparagrafi.

caption serve per personalizzare la composizione delle didascalie.

geometry serve per personalizzare il layout della pagina, la sua geometria, secondo schemi tradizionali o moderni, sempre parametrizzati, in modo da essere completamente liberi nella definizione dei margini, dei contrografismi fra testatina o piedino e il corpo della pagina; per scegliere la giustezza in modo da avere il numero (medio) ottimale di caratteri in ogni riga in relazione al font usato e al suo corpo 'normale'. Sotto molti aspetti ***geometry*** agisce come il pacchetto ***typearea***, apparentemente molto popolare fra gli esperti che scrivono sul forum GUIT, ma è decisamente più agile e consente di personalizzare in modo più esteso.

Ovviamente questi pacchetti sono utili anche nel creare nuove classi, perché basta caricarli all'inizio della nuova classe, specificando loro i parametri desiderati, così che spesso la creazione di una nuova classe si riduce veramente a poco.

6.4 Alcune classi non standard

Il sistema TEX viene distribuito con una enorme quantità di file aggiuntivi, compresi i file di classe che alcuni autori hanno predisposto e messo a disposizione degli utenti.

6.4.1 Le classi Komascript

Markus Kohm (da cui il prefisso 'Koma') ha messo a disposizione le quattro classi fondamentali per la composizione di libri, rapporti, articoli e lettere,

rispettivamente *scrbook*, *scrreprt*, *scrartcl* e *scrlttr2*. Le caratteristiche e i comandi disponibili sono reperibili nella documentazione del pacchetto *koma-script*.

Ma la componente di questo pacchetto certamente più utile è il pacchetto *typearea*, richiamabile esplicitamente dai file di classe, che permette di disegnarsi a proprio piacimento la griglia di scrittura secondo i criteri più diffusi; il pacchetto *typearea* è richiamabile anche da altri file di classe, non è una esclusiva della collezione Komascript di Markus Kohm.

In questa collezione di classi, la classe *scrlttr2* è molto interessante perché descrive il layout delle lettere in una maniera più adatta alle tradizioni europee; una caratteristica particolare è la sua capacità di adattare il formato della carta e quello della busta, persino delle buste con finestra trasparente, molto usate nella corrispondenza burocratica e commerciale. La documentazione spiega anche come predisporre nuove opzioni e/o configurazioni locali adatte a formati di carte e di buste non conformi alle norme ISO, anche nella versione molto dettagliata delle norme ISO adattate al Giappone. Per quel che riguarda l'Italia, esistono delle norme relative all'abbinamento della carta con la busta, ma apparentemente non esistono delle norme per le buste con finestra.

Le classi della collezione Komascript sono alla base anche di classi o di pacchetti molto in voga fra i frequentatori del forum del GuIT, in particolare per il pacchetto *ClassicThesis*. Questo fatto discende anche dalla facilità con cui è possibile configurare il layout della pagina e gli stili compositivi di tutte le parti di sezionamento, dai titoli correnti di ogni livello, alle didascalie, dalle testatine ai piedini.

Attenzione: lavorare con le classi Komascript vuol dire confrontarsi con un insieme di file di estensione molto potente, ma commentato bene solo in tedesco; la traduzione in inglese, disponibile nelle distribuzioni del sistema TEX, lascia un poco a desiderare. Per cui è difficile sfruttare appieno le funzionalità di questi file di estensione, compreso l'ottimo pacchetto *typearea*.

Viceversa chi scrive trova molto interessante il pacchetto *scrextend* che permette di definire ad un valore arbitrario il corpo normale da usare in una classe, e di ridefinire gli altri comandi per la scelta del corpo di font in modo che seguano sostanzialmente la stessa progressione che questi corpi hanno con le classi standard; se poniamo unitario il corpo normale, gli altri corpi corrispondono alla sequenza 0,5, 0,6, 0,7, 0,8, 0,9, 1,0, 1,2, 1,44, 1,728, 2,074, 2,488, 2,986, 3,583. Perciò se con questo pacchetto si dichiara che il corpo normale vale, per esempio, 11,5 pt, gli altri corpi sono ottenuti moltiplicando questo valore per i coefficienti della suddetta sequenza. Va da sé che questo modo di procedere funziona solo con font vettoriali continuamente scalabili. Si noti anche che tutti i contrografismi vengono proporzionali al corpo normale e adattati agli altri corpi in proporzione. Ci si può domandare a cosa serva regolare il corpo normale con la finezza di frazioni di punto tipografico, ma per l'esperienza acquisita da chi scrive la cosa diventa importantissima per testi di pregio e/o destinati ad usi particolari.

6.4.2 La classe *memoir*

Peter Wilson ha scritto questa unica classe con la quale è possibile comporre quasi tutto quello che si può comporre con le classi standard, ma offre possibilità di personalizzazione difficili da realizzare se ci si appoggia alle classi standard.

Una parte pregevole di questo lavoro è costituita dal file contenente il manuale di documentazione, assai lungo e dettagliato, contenente anche una buona dose di consigli stilistici e una discreta storia del disegno grafico del libro e del documento[1]; per le parti tecniche il manuale è essenziale perché l'estensione di LaTeX e delle sue classi che la classe *memoir* offre è decisamente importante.

La classe permette di personalizzare qualunque aspetto del documento, dal disegno grafico della pagina, alla scelta dei font per scrivere qualunque testo speciale, compresi i titolini delle sezioni, le testatine e i piedini. Qualunque cosa possa essere personalizzata, e che è difficile personalizzare usando LaTeX standard, con *memoir* può essere fatta facilmente e con poche istruzioni.

L'utente che si rivolge a *memoir* finisce con restarci affezionato per sempre; ma deve fare molta attenzione con le personalizzazioni, perché queste sono utili, certamente, ma come si può migliorare qualunque cosa, la si può anche peggiorare. In altre parole questa classe è decisamente più adatta ad un utente esperto, cosa che riesce a chiunque abbia un minimo di sensibilità estetica ed abbia spirito di osservazione per imparare e valutare i pregi e i difetti di ciò che viene fatto da altri. Dagli altri si dovrebbe prima di tutto imparare ad evitare gli errori, cercando di imitare il meglio e di avvicinarsi il più possibile alle realizzazioni degli artisti grafici più quotati; con lo strumento *memoir* ci si può riuscire.

L'autore di *memoir* nel suo documento `memdesign` racconta cose che hanno a che vedere con la cultura tipografica oltre che con il book design. Vale la pena di citare questo brano tradotto in italiano da Luciano Battaia.

> L'essenza di un libro ben stampato è che non si fa notare al primo, o addirittura al secondo o successivo, sguardo di chiunque non abbia un occhio allenato. Se la vostra prima reazione nello sfogliare un libro è di fare un'esclamazione di meraviglia osservando il layout, allora il libro è molto probabilmente mal progettato, se mai è stato progettato. La stampa di qualità è raffinata, non stridente.

[1] Questo, in verità, appariva sul manuale di *memoir* fino a qualche anno fa; ora il manuale è molto più tecnico e chiarisce ogni possibile dubbio di programmazione, ma in verità quell'excursus storico era molto interessante. L'excursus storico è ancora disponibile come file a parte ed è leggibile con il comando da terminale `texdoc memdesign`. L'autore, Peter Wilson scrive: "These notes briefly cover some aspects of book design and typography, independently of the means of typesetting. Among the several books on the subject listed in the Bibliography I prefer Bringhurst's *The Elements of Typographic Style* (Bringhurst 1999).
The notes originally formed the first part of a user manual for the memoir class for use with the LaTeX typesetting system developed by Leslie Lamport (Lamport 1994) based on Donald Knuth's TeX system (Knuth 1984). The manual was first published in 2001 and as the notes have grown in size and memoir's capabilities have been extended the manual also grew to approaching 700 pages (Wilson 2009). At that point seemed advantageous to separate the design notes from the technicalities, hence this document."

> Con l'avvento del desktop publishing molti autori hanno la tentazione di progettare da soli i loro testi. Sembra molto facile farlo. Basta scegliere alcune delle migliaia di font disponibili, usarne uno per i titoli, uno per il testo principale, un altro per le didascalie, decidere le dimensioni dei caratteri, e la cosa è fatta.
>
> Tuttavia, come scrivere è un'abilità che bisogna apprendere, anche comporre tipograficamente un testo è un'arte che si deve apprendere e su cui bisogna esercitarsi. Ci sono centinaia di anni di esperienza racchiusi nel buon design di un libro. Essi non possono essere trascurati con leggerezza e molti autori che progettano i loro libri non conoscono alcune delle conquiste più importanti, per non parlare del fatto che quello che fanno è esattamente in antitesi con esse. Un esperto può infrangere le regole, ma allora sa che ha delle buone ragioni per farlo.
>
> [...] Se un libro grida 'guardami', questo è un segnale, e un pessimo segnale, per chi l'ha progettato.

6.4.3 La classe *ncc*

Alexander Rozhenko ha predisposto una classe *ncc*, valida per tutte le lingue accessibili con *babel*, ma pensata inizialmente per il russo; questa classe permette di comporre articoli, rapporti e libri semplicemente specificando opportune opzioni a questa unica classe. La classe è accompagnata da un certo numero di file di estensione che consentono di eseguire composizioni particolari; l'autore non le ha considerate di uso generale, cosicché ha potuto mantenere le dimensioni della classe entro limiti ragionevoli, ma ha consentito di raggiungere prestazioni supplementari con moduli aggiuntivi da usarsi di volta in volta a seconda delle necessità compositive.

La classe è estremamente versatile e consente di unire le prestazioni di molte estensioni LaTeX in un'unica classe. Cito a caso gli allineamenti di equazioni, quelli di *amsmath* tanto per intenderci, realizzati con pochi comandi assai versatili; gli oggetti flottanti 'mini', per minifigure e minitabelle, comprese le (piccole) figure e tabelle a margine del testo; vari tipi di teoremi già predisposti, senza bisogno di ricorrere a `\newtheorem`; ovviamente ci sono anche i comandi per le dimostrazioni.

Tutti i comandi dispongono di parametri facoltativi che consentono una personalizzazione piuttosto ampia.

La documentazione è abbastanza dettagliata e si trova in `.../doc/latex/ncclatex/ncclatex.pdf`; è opportuno studiare bene la documentazione, perché, sebbene si tratti sempre di mark-up LaTeX, esistono moltissimi nuovi comandi e molti ambienti sono sostituiti da comandi con diversi argomenti in parte facoltativi e in parte obbligatori. Bisogna, insomma, prenderci la mano, ma ne vale la pena.

6.4.4 Le classi per le tesi di laurea

Spesso gli studenti si avvicinano a LaTeX quando devono scrivere la tesi di laurea. Dagli archivi CTAN si possono scaricare diversi pacchetti che contengono il necessario per comporre la tesi di laurea o di dottorato. Sono disponibili sia file di classe destinati allo scopo, sia pacchetti che servono per estendere alcune classi generiche alla composizione delle tesi; esistono già anche dei modelli (*template*) che rappresentano delle "intelaiature" già pronte per impostare la tesi e che già caricano le classi o i pacchetti necessari. Qui si citano i modelli *TesiClassica* [51], *TesiModerna* [52]; i pacchetti *ClassicThesis* [42] e il file di estensione `toptesi.sty`; le classi *TOPtesi* [5], *sapthesis* [10], *suftesi* [69]; i file di estensione *frontespizio* [29] e `topfront.sty` servono invece solo per comporre il frontespizio della tesi. Le classi, i pacchetti e i file citati sono stati quasi tutti scritti da italiani per gli studenti universitari italiani.

ClassicThesis scritta da un docente tedesco, ma adatta a tutte le lingue, offre un bellissimo design della pagina, che sarebbe quanto mai indesiderabile personalizzare, perché si perderebbe tutto il bello di questo pacchetto. Siccome il layout della pagina è piuttosto originale, può non adattarsi alle specifiche di questa o quella università.

Anche *TesiClassica* che si appoggia a *ClassicThesis* è più sobria, ma è difficile da personalizzare. *TesiModerna* invece è più tradizionale e non si appoggia a *ClassicThesis*, tuttavia qualunque tipo di personalizzazione di questi due modelli può rovinarne il layout molto bello.

sapthesis e *suftesi* permettono diverse opzioni per i layout della pagina ma, eseguita questa scelta, sarebbe meglio non metterci più le mani per personalizzare ulteriormente. Va notato che fra le opzioni di *suftesi* ce ne sono diverse per produrre articoli e libri in formati diversi dal solito formato di carta A4; uno di questi formati e stili di composizione imita molto bene quello usato da Robert Bringhurst[2] per comporre il suo testo *The Elements of Typographic Style* [11]. *suftesi* è fortemente personalizzabile, ma non nel disegno della pagina, per cui potrebbe essere incompatibile con le specifiche richieste di alcune università. *unifith* mette a disposizione una classe per tesi magistrali e dottorali secondo i requisiti dell'università di Firenze come, d'altra parte *sapthesis* mette a disposizione una analoga classe per le tesi da comporre all'università La Sapienza di Roma.

Il pacchetto *frontespizio* di Enrico Gregorio si dedica esclusivamente al frontespizio della tesi; questo è completamente configurabile in ogni suo dettaglio, per cui è possibile predisporre il frontespizio della tesi virtualmente per ogni prescrizione di segreteria di facoltà e/o per ogni ateneo. L'unica cosa a cui bisogna fare attenzione è che questo bellissimo pacchetto *non* consente di fare le 'porcherie' richieste in alcune facoltà; evidentemente chi ha redatto quelle

[2] L'autore di *suftesi*, Ivan Valbusa, ha informato chi scrive che l'opzione *elements*, che serviva per imitare lo stile di Bringhurst, non è più disponibile a partire dall'ultimo aggiornamento del 2016. Peccato: era un bello stile, ma chi scrive comprende il perché di questa decisione, visto che era molto diverso dagli stili 'normali'.

prescrizioni o non è ancora passato al calcolatore e usa ancora la macchina da scrivere, o usa solamente e male un word processor (che usato bene produrrebbe anche risultati buoni), oppure ignora completamente i rudimenti della tipografia. Gli studenti universitari che si accingono a scrivere la tesi non si scoraggino: se una cosa non si può fare con *frontespizio* allora vuol dire che è meglio non farla!

Il pacchetto *TOPtesi* contiene sia il file di classe, sia un pacchetto omonimo con cui si può configurare un certo numero di classi standard; contiene un pacchetto con comandi utili, che può essere usato indipendentemente dalla classe, e un pacchetto per il frontespizio, con il quale probabilmente si possono personalizzare diverse classi; la documentazione di *TOPtesi* spiega come usare il modulo per il frontespizio con qualunque altra classe standard.

Il pacchetto *TOPtesi* è predisposta per comporre tesi in italiano, in inglese (anche il frontespizio) e in francese e come lo si può adattare ad altre lingue; per comporre il frontespizio in lingue diverse dalle due citate, si possono usare comandi specifici che possono essere inseriti in un file di configurazione ed è stata usata per scrivere tesi in francese, spagnolo, tedesco, visto che uno degli obbiettivi dell'autore era quello di mettere a disposizione un pacchetto in grado di soddisfare le esigenze degli studenti in mobilità Erasmus; perciò il pacchetto è personalizzabile per ogni lingua e per molti stili universitari. Il pacchetto è stato pensato anche per scrivere le tesi completamente in lingue diverse dall'italiano e dall'inglese in vista del fatto che gli studenti in doppia laurea con i programmi Erasmus devono scrivere la tesi anche (o solo) nella lingua dell'università ospitante.

La classe *topesi*, elemento chiave del pacchetto *TOPtesi*, è nata come una sovrastruttura della classe *report* ma, usando solo il suo file di estensione `toptesi.sty`[3], non la classe, funziona anche con la classe *book*; la classe *toptesi*, benché costruita sopra la classe *report*, che non distingue le tre parti di un documento complesso, fa invece questa distinzione automaticamente: essa comincia a comporre con lo stile della `\frontmatter` come nella classe *book*, ma nel momento in cui viene eseguito il suo comando `\indici`, vengono composti i vari indici richiesti e poi viene eseguito l'equivalente del comando `\mainmatter`; con il comando `\appendix`, di fatto si passa automaticamente alla composizione del materiale finale, ma se non ci sono appendici è necessario esplicitare il passaggio alla parte finale mediante il comando `\backmatter`.

Nella classe *toptesi* le appendici sono forzatamente nella parte finale, ma vengono identificate con "numeri" costituiti dalle lettere maiuscole. Invece nella *book*, il comando `\backmatter` elimina la dicitura 'Appendice ⟨*lettera*⟩' prima

[3]Può sembrare molto confuso, ma con packTOPtesi si indica una grossa collezione di file che in inglese si chiama *bundle*, ma manca di un nome specifico in italiano; la classe *toptesi* è costituita dal file `toptesi.cls` e il suo nome senza estensione costituisce l'argomento obbligatorio del comando `\documentclass`; il modulo di estensione *toptesi.sty*, che in inglese si chiama *package* che la classe *toptesi* immette con il comando `\RequirePackage`, o l'autore immette nel preambolo della sua tesi se non usa la classe omonima, è formato dalla collezione di macro che permettono di comporre una qualsiasi tesi. Dalla versione 6.0.00 in poi la collezione *TOPtesi* usa moduli di estensione specifici per ogni tipologia di tesi.

del titolo di ogni appendice e le compone come capitoli non numerati. Questa classe, lo si ripete, prevede che le appendici facciano parte del materiale finale, ma le tratta come normali capitoli numerati (con lettere maiuscole). Ovviamente la questione se le appendici facciano o non facciano parte del materiale finale è una questione di gusti e di tradizione tipografica; il grande e citatissimo 'book designer' Bringhurst attribuisce le appendici al materiale finale. Chi ha predisposto la classe `book` attribuisce, invece, le appendici al materiale centrale del documento.

Molti dei pacchetti citati sono usabili anche con xelatex e lualatex; questo è particolarmente utile per tesi che trattino di lingue scritte in alfabeti diversi da quello latino o anche quando contengano una simbologia molto ricca.

6.4.5 L'estensione `layaureo`

Fabiano Busdraghi ha prodotto un pacchetto di estensione che non modifica gli altri parametri della classe in uso, ma imposta il disegno grafico della pagina secondo il criterio della sezione aurea; il pacchetto si chiama `layaureo` e, nel modo più semplice possibile, senza ricorrere ad impostazioni di parametri o di parole chiave talvolta 'immaginose', modifica la larghezza del testo e la sua altezza, nonché i margini di pagina in modo da ottenere il disegno grafico basato sulla sezione aurea.

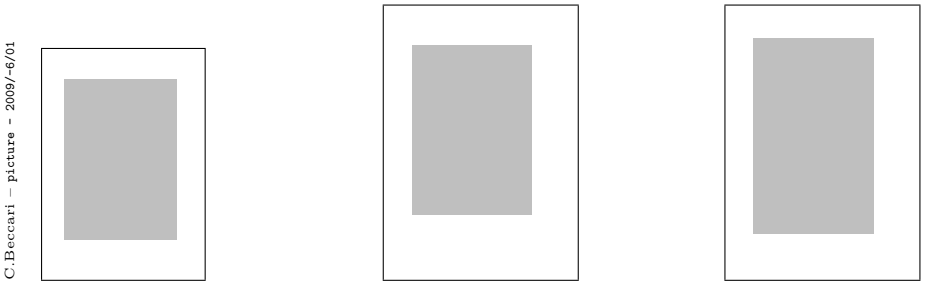
In particolare allarga la giustezza in modo da coprire meglio la pagina A4 di quanto non lo facciano le classi standard; questo allargamento dipende dal corpo normale del font in uso. Poi se il documento deve essere composto sul recto e sul verso delle pagine, allora sistema il blocco del testo in modo che i margini stiano fra loro come la sezione aurea. È possibile usare una opzione per spostare tutto il testo di una quantità fissa verso l'esterno, per esempio 5 mm, al fine di tenere conto della rilegatura.

Si ricorda che la sezione aurea è un rapporto usato molto spesso nelle arti grafiche e lo si ritrova in moltissime strutture naturali; per la sua definizione si veda in particolare l'equazione etichettata (sezione aurea) nella pagina 304.

Senza tenere conto dell'aumento eventuale del margine interno per tenere conto della rilegatura, la scelta della sezione aurea implica, per esempio, che il margine interno sia la frazione 0,618... del margine esterno.

Vale la pena di mettere a confronto tre layout di pagina nella figura 6.1; il primo layout corrisponde esattamente a questo testo composto con la classe `book` alla quale non sono state apportate modifiche geometriche di nessun genere.

Il secondo layout ha la pagina, il testo e i margini in rapporto aureo fra di loro; per essere più precisi, il rettangolo della pagina rappresenta in scala il foglio UNI A4, i cui lati hanno un rapporto di $\sqrt{2}$ fra di loro e non è quindi un rettangolo aureo; però la larghezza della pagina e la giustezza del testo sono in rapporto aureo, così come lo sono i margini esterno ed interno fra di loro; gli stessi rapporti valgono per le dimensioni verticali; a causa di questa regolarità delle proporzioni si può osservare che la diagonale NW–SE della gabbia del testo coincide con la diagonale della pagina. Si noti anche che la gabbia del testo è un



(*a*) (*b*) (*c*)

Figura 6.1: Tre geometrie di pagina a confronto. (*a*) Gabbia (in grigio) e margini per questo testo composto con la classe `book` su foglio in formato B5; (*b*) gabbia e margini in rapporto aureo fra di loro su su foglio in formato A4; (*c*) gabbia e margini ottenuti con l'estensione `layaureo` su foglio in formato A4.

rettangolo UNI, cioè con il rapporto $\sqrt{2}$ fra l'altezza e la base; esso non ha le proporzioni auree.

Il terzo layout è quello che si ottiene con l'estensione `layaureo` sulla pagina UNI A4 per la quale è stato disegnato; la giustezza è fissata circa 10 mm più grande della giustezza di default della classe `book` ma non è in rapporto aureo con la larghezza della pagina; invece i margini interno ed esterno lo sono fra di loro; i margini superiore e inferiore, al contrario, sono in rapporto di $\sqrt{2}$ e il margine inferiore è il più grande. L'altezza della gabbia è in rapporto aureo con la sua base e forma un rettangolo aureo.

A causa dei margini ristretti, in particolare quello esterno, i layout basati in tutto o in parte sulla sezione aurea non sono adatti ad accogliere le note marginali composte a pacchetto. La classe `book` usata con una pagina in formato A4, invece che B5 come in questo testo, ha margini esterni abbastanza generosi da poter accogliere anche le note marginali.

La figura 6.2 mostra chiaramente la differenza fra il rettangolo UNI, basato sulla geometria del quadrato, e il rettangolo aureo, basato sulla geometria del pentagono regolare; in entrambi i casi l'altezza del rettangolo è pari alla corda maggiore del poligono regolare avente la base uguale a quella del rettangolo, ma, a pari base, il rettangolo aureo, figura 6.2(*b*), è significativamente più alto del rettangolo UNI, figura 6.2(*a*).

Nella pagina 170 e nella successiva sono mostrate in dettaglio tutte le misure che riguardano le pagine pari e dispari di questo testo. Le misure indicate in punti tipografici sono state ricalcolate da LaTeX stesso, caso mai fossero state modificate in qualche modo; in realtà non è stato modificato nulla rispetto alle impostazioni di default.

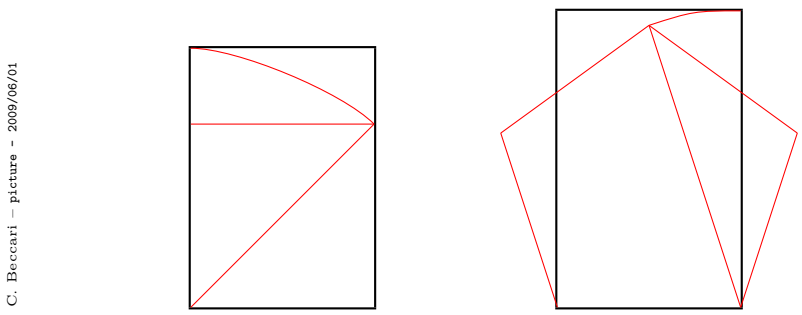


Figura 6.2: Costruzioni geometriche per mettere in relazione l'altezza del rettangolo con il lato del poligono regolare sul quale è costruito. (*a*) rettangolo ISO; (*b*) rettangolo aureo.

6.5 I pacchetti di estensione

Anche nel paragrafo precedente si è parlato di alcuni pacchetti di estensione; virtualmente se ne parla in ogni capitolo.

I pacchetti di estensione, spesso chiamati *file di stile* a causa dell'estensione `.sty` che essi hanno, estendono le capacità di LaTeX mettendo a disposizione del compositore nuovi comandi o nuovi stili di composizione.

6.5.1 Come invocare i file di estensione

Nel preambolo di questo testo sono stati invocati diversi pacchetti di estensione; di alcuni si è già detto nel capitolo 5. Mentre i pacchetti descritti in quel capitolo sono da usarsi virtualmente ogni volta che si scrive un documento scritto con una tastiera nazionale (probabilmente in Italia con una tastiera italiana), con un alfabeto latino che contiene lettere accentate e in una lingua diversa dall'inglese, ora si mostrano le invocazioni degli altri pacchetti richiesti per la composizione di questo specifico testo:

```
\usepackage{curve2e}[2019-01-01]
\usepackage{graphicx}
\usepackage{mflogo}
\usepackage{amsmath,amssymb,amscd}
\usepackage{afterpage}
```

Il comando di invocazione è `\usepackage` che segue la sintassi seguente:

`\usepackage[`⟨*opzioni*⟩`]{`⟨*pacchetto*⟩`}[`⟨*data*⟩`]`

Le ⟨*opzioni*⟩ sono quelle specifiche del pacchetto invocato; per esempio, per `babel` il pacchetto era stato invocato nel preambolo con il comando

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

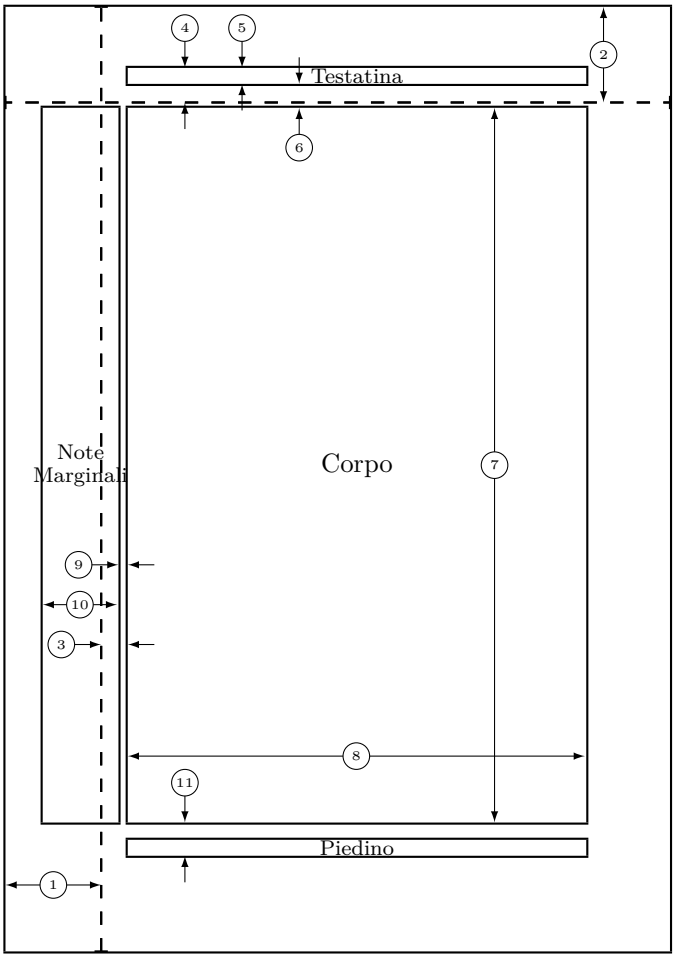
Il ⟨*pacchetto*⟩ può consistere nel nome di un solo file senza la specificazione dell'estensione `.sty`, ma può anche consistere in una lista di nomi separati l'uno dall'altro da una virgola. La ⟨*data*⟩ è la data scritta secondo le norme ISO nella forma *aaaa-mm-gg* (anno con tutte quattro le cifre, mese indicato con due cifre anche se la prima è uno zero, giorno indicato con due cifre anche se la prima è uno zero) che indica la data rispetto alla quale quella del pacchetto deve essere uguale o posteriore; questa indicazione serve per essere sicuri di usare pacchetti sufficientemente aggiornati; come si vede nell'esempio riportato sopra, si è richiesto il pacchetto *curve2e* (che a sua volta carica il pacchetto *pic2e*) di data successiva o uguale al 1° gennaio 2019, perché le versioni precedenti a questa data non disponevano di certi nuovi comandi introdotti in un secondo tempo.

Si fa notare che le ⟨*opzioni*⟩ indicate nell'invocazione del pacchetto sono *locali* a quel pacchetto e se questo non le riconosce viene emesso un avvertimento che informa che l'opzione tale-e-tale non è stata riconosciuta. Invece le ⟨*opzioni*⟩ specificate nella invocazione del file di classe attraverso il comando `\documentclass` sono *globali*, nel senso che esse vengono passate anche alle successive invocazioni dei pacchetti; l'opzione *italian* specificata per *babel* avrebbe potuto essere indicata fra le opzioni di `\documentclass` e così avrebbe potuto essere usata, senza bisogno di ripeterla, anche da altri pacchetti, come per esempio *varioref* o *layout*; il primo esegue le citazioni nella forma, per esempio, '... nella figura "3.5 della pagina 123"...'; il secondo pacchetto serve per disegnare in una pagina a se stante il layout grafico della pagina, indicando graficamente i rettangoli che descrivono la gabbia, o la testatina e, a parole, le indicazioni metriche che nel disegno sono riportate solo mediante delle frecce. È appunto il pacchetto che è stato usato per disegnare i layout della pagina 170 e della successiva.

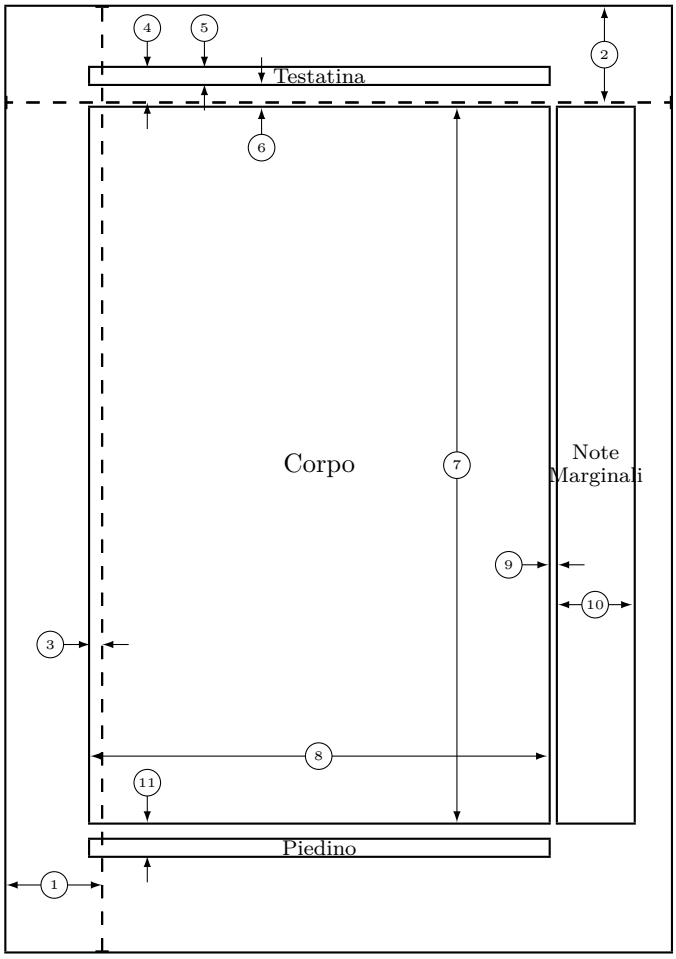
6.5.2 I vari pacchetti e gli archivi internazionali

È facile comprendere che i pacchetti disponibili siano numerosissimi; l'archivio internazionale CTAN (che sta per Comprehensive TEX Archive Network) ne contiene diverse migliaia ottenuti come contributi dei vari utenti di LATEX; come succede con il software libero, gli utenti hanno realizzato qualche estensione per risolvere qualche loro problema contingente e hanno ritenuto che fosse utile anche per gli altri utenti; perciò hanno caricato il pacchetto nell'archivio in modo che chiunque potesse servirsene liberamente e gratuitamente.

Il sistema degli archivi CTAN è formato da alcuni archivi principali, uno negli USA e uno in Germania, che sono sempre sincronizzati l'uno con l'altro e costituiscono la sorgente principale delle estensioni e delle distribuzioni gratuite del sistema TEX. Sparsi per il mondo ci sono poi innumerevoli siti che costituiscono i *mirror* degli archivi principali; questi mirror vengono sincronizzati dai loro amministratori quasi in tempo reale, ma generalmente non sono in ritardo al più di un paio di giorni rispetto a quelli principali. Per questo motivo c'è sempre la



1 un pollice + \hoffset
2 un pollice + \voffset
3 \evensidemargin = 20pt
4 \topmargin = -26pt
5 \headheight = 12pt
6 \headsep = 18pt
7 \textheight = 538pt
8 \textwidth = 345pt
9 \marginparsep = 7pt
10 \marginparwidth = 57pt
\marginparpush = 5pt (non mostrato)
11 \footskip = 25pt
\hoffset = 0pt
\voffset = 0pt
\paperwidth = 500pt
\paperheight = 711pt



1	un pollice + \hoffset	2	un pollice + \voffset
3	\oddsidemargin = -9pt	4	\topmargin = -26pt
5	\headheight = 12pt	6	\headsep = 18pt
7	\textheight = 538pt	8	\textwidth = 345pt
9	\marginparsep = 7pt	10	\marginparwidth = 57pt
11	\footskip = 25pt		\marginparpush = 5pt (non mostrato)
	\hoffset = 0pt		\voffset = 0pt
	\paperwidth = 500pt		\paperheight = 711pt

possibilità di avere un archivio abbastanza vicino così che le connessioni internet siano le più rapide possibile.

Tutti gli archivi e quasi tutti i mirror hanno una pagina interattiva che permette di cercare o per nome o per argomento i file che interessano; solitamente sono anche predisposti per permettere di scaricare intere cartelle in formato compresso così da essere sicuri di non dimenticare nulla per la strada.

Le distribuzioni migliori dispongono di un wizard, altre di un assistant, altre di un package manager, che permettono di avere preliminarmente il nome di tutti i pacchetti disponibili e di scaricare sul proprio calcolatore solo i pacchetti che interessano.

Gli archivi principali sono:

```
dante.ctan.org
tug.ctan.org
```

In Italia c'è il mirror:

```
ctan.mirror.garr.it
```

ma, lingua permettendo, si può accedere ai mirror francesi, svizzeri, austriaci e, in particolare all'archivio dell'associazione Dante (`http://www.dante.de`), localizzato nella Germania meridionale; il mirror localizzato in Svizzera all'indirizzo `ftp://mirror.switch.ch/mirror/tex`, o quello localizzato in Austria all'indirizzo `http://gd.tuwien.ac.at/publishing/tex/CTAN/` sono particolarmente efficaci. Nel seguito con CTAN si indicherà indifferentemente l'insieme degli archivi internazionali o semplicemente l'indirizzo di base di uno degli archivi principali o di un mirror.

È bene dirlo subito: normalmente non è necessario collegarsi a nessun archivio principale o 'specchiato'; è invece importante sapere come configurare il proprio package manager, o wizard, o assistant per potersi collegare quando è necessario aggiornare la propria installazione completa; questa è una operazione di configurazione da farsi una volta per sempre e poi non ci si pensa più; ci si collega senza dover specificare nessun indirizzo internet, ci si aggiorna, e il gioco è fatto. È più importante collegarsi al sito ufficiale `www.tug.org` dell'associazione internazionale degli utenti di TEX (TEX Users Group), dove si possono trovare notizie interessanti, situazione di aggiornamento degli archivi, documentazione, DVD, notizie sulle conferenze internazionali e nazionali.

6.6 Come scrivere nuovi pacchetti

Cercando bene si può quasi sempre trovare il problema già risolto in un qualche pacchetto presente negli archivi.

Tuttavia talvolta può essere necessario crearsi un pacchetto di estensione personale; per questo scopo bisogna leggere con attenzione la già citata guida

`clsguide` nella cartella del proprio sistema dedicata alla documentazione `.../doc/latex/base/`; si scoprirà che la cosa di per sé è facilissima; se c'è qualche difficoltà questa risiede nella scelta di che cosa mettere nel pacchetto personale.

C'è però una risposta semplicissima; tutti hanno bisogno di estensioni personali e prima o poi si scrivono dei nuovi comandi per il proprio uso, che rispecchiano il proprio stile di gestione delcodice da inserire nei file sorgente. Possono essere macro per inserire certi loghi o certi marchi che hanno a che fare con l'istituzione alla quale si appartiene; possono essere macro per scrivere sigle o abbreviazioni frequenti nei testi che si scrivono abitualmente; possono essere nuovi ambienti che riflettono le proprie necessità compositive; possono essere anche semplici modifiche di comandi standard.

Bene, tutto ciò può essere collocato in un file chiamato in una maniera un po' personalizzata, diversamente dal solito `myfile.sty`, per esempio `estensionedi`⟨*iniziali*⟩`.sty`, dove ⟨*iniziali*⟩ può consistere nelle iniziali del proprio nome e cognome o quant'altro personalizzi il proprio file. L'estensione del nome del file *deve* essere `.sty`.

Seguendo le istruzioni della Guida, il file deve cominciare con due righe importantissime:

```
\NeedsTeXFormat{⟨formato⟩}[⟨data⟩]
\ProvidesPackage{⟨nome-del-pacchetto⟩}[⟨data, versione e descrizione⟩]
```

Il ⟨*formato*⟩ se usate LaTeX 2ε, come si suppone, sarà `LaTeX2e`; la ⟨*data*⟩ sarà la data del formato che voi avete usato per scrivere le vostre macro; l'indicazione di questa data serve per assicuravi che il vostro file per sbaglio non possa venire usato da voi (su un'altra macchina) o da altri con una versione obsoleta del ⟨*formato*⟩; la data, scritta nella forma ISO, cioè *aaaa-mm-gg* (oppure*aaaa/mm/gg*), può essere indicata opzionalmente anche nella riga nella quale si richiama con `\usepackage` un pacchetto qualsiasi, in modo da essere sicuri di non usare versioni obsolete; naturalmente bisogna conoscere la data del pacchetto non obsoleto che vi serve.

Il ⟨*nome-del-pacchetto*⟩ sarà verosimilmente il nome del vostro file personale, mentre la ⟨*data, versione e descrizione*⟩ deve essere una stringa di parole, possibilmente di lunghezza non maggiore di una riga di caratteri della finestra comandi, o del terminal, o della console; queste informazioni devono avere la forma seguente:

⟨*aaaa-mm-gg*⟩ `v.`⟨*versione*⟩ ⟨*breve descrizione*⟩

La data, in formato ISO, è quella della creazione del pacchetto o della sua ultima revisione. La versione si può scrivere nella maniera preferita. La breve descrizione deve essere una breve frase del tipo "Macro di Mario Rossi per comporre il manuale di tricotetratomia".

Dopo queste due righe si possono inserire tutte le proprie definizioni e/o richiamare con `\RequirePackage`i pacchetti necessari; è bene terminare il file con `\endinput`.

L'argomento delle definizioni verrà trattato nel capitolo 19.

6.7 Non modificare i pacchetti esistenti

Alcuni utenti di LaTeX credono di poter risolvere i loro problemi compositivi andando a modificare i file di classe o i pacchetti presenti nella propria distribuzione del sistema TeX o scaricati da CTAN.

Questo deve essere assolutamente evitato! Non lo si ripeterà mai abbastanza, ma questa pratica, oltre ad essere vietata dalla licenza a cui sono sottoposti quasi tutti i pacchetti esistenti, è una pratica autolesionista.

I modi corretti di procedere sono i seguenti.

1. Se le modifiche da apportare alla classe o al file di stile sono numerose, il modo migliore di procedere è quello di copiare il file in una cartella dell'albero personale `.../tex/latex/`⟨*cartella*⟩. Il nome di questa ⟨*cartella*⟩ verrà scelto in modo mnemonico, ma diverso dai nomi delle cartelle già esistenti; il file copiato verrà ribattezzato con un altro nome; per esempio se si vuole modificare la classe *book* si copierà il file `book.cls` nella propria ⟨*cartella*⟩ dandole, per esempio, il nome `tttomia.cls`.
   In questo nuovo file si cambi il contenuto di `\ProvidesClass`, per esempio modificando la dichiarazione

   ```
   \ProvidesClass{book}
                 [2004/02/16 v1.4f Standard LaTeX document class]
   ```

   nel modo seguente

   ```
   \ProvidesClass{tttomia}
                 [2017/03/10 v1.0
            Mario Rossi: classe per il libro di TricoTetraTomia]
   ```

   In questo nuovo file si apportino tutte le modifiche che si desiderano e vi si aggiungano tutti i comandi che si vogliono usare; questo si è liberi di farlo secondo la licenza delle classi, ma non si è liberi di modificare il file originale senza cambiargli il nome.
   Nello stesso tempo si può usare la classe standard in qualunque altra circostanza; questa inoltre potrà servire per seguire la stessa strada, con nomi diversi, se si vuole adattare la classe *book* alla composizione di un altro libro.
   Lo stesso procedimento può essere seguito per modificare i file di estensione.
2. In alternativa si può modificare solo qualche comando che interessa in modo particolare; si copiano dal file di classe originale o dal file di estensione i comandi che si vogliono modificare e li si incollano in un proprio file `mymacros.sty` di comandi e macro personali; si sostituisce il comando `\newcommand` con `\renewcommand` facendo operazioni simili per le definizioni degli ambienti. Nel corpo della definizione si inseriscono tutte le modifiche che si vogliono.

3. Esistono modi più avanzati per eseguire modifiche puntuali a determinati comandi. Basta usare il pacchetto *xpatch* e usare i suoi comandi applicabili anche ai comandi originali di classi e pacchetti, senza bisogno di ricopiarli nel proprio file di macro personali. Questa strada è delicata e riservata ad utenti esperti, ma permette di fare correzioni personali in modo estremamente efficace e in maniera da non toccare assolutamente i file originali. In ogni caso per apportare qualunque modifica bisogna conoscere le definizioni da modificare, altrimenti si rischia di mettere il *patch* nel posto sbagliato.

Ci si ricordi che i comandi definiti dentro i file di classe o di estensione originali contengono spessissimo la 'lettera' @; è lecito usare questo segno come se fosse una lettera dell'alfabeto solo all'interno quei file di classe o di estensione; quindi anche all'interno del proprio file `mymacros.sty`. Si stia attenti però a non combinare pasticci; modificare o ridefinire comandi che contengano @ nel loro nome espone al rischio di modificare un comando di sistema in modo errato e tutto il sistema TEX potrebbe diventare inutilizzabile; la riparazione è semplicissima, perché vi basta tornare sui propri passi eliminando dal proprio file le modifiche o le definizioni che potrebbero avere sconvolto il sistema, e poi si può ricominciare con più attenzione stando bene attenti a che cosa si ridefinisce o si modifica. Si usino sempre i comandi `\newcommand` e `\renewcommand` che controllano la precedente esistenza di comandi con lo stesso nome; si possono usare per queste verifiche anche i comandi di sistema `\@ifdefinable` e `\@ifundefined`; le rispettive sintassi sono le seguenti:

```
\@ifdefinable{⟨comando⟩}{⟨azione⟩}
\@ifundefined{⟨nome del comando⟩}{⟨azione⟩}
```

Il comando `\@ifdefinable` verifica che il ⟨*comando*⟩ che si vorrebbe definire sia definibile; cioè che (*a*) non sia stato già definito, (*b*) sia diverso da `\relax`, (*c*) non cominci con `\end`. Se sono verificate queste condizioni, allora si può eseguire l'⟨*azione*⟩, che verosimilmente consiste nella definizione vera e propria del ⟨*comando*⟩.

Invece `\@ifundefined` verifica che il ⟨*nome del comando*⟩ (il comando privato dell'iniziale backslash) sia davvero una entità priva di definizione, e se è verificata questa condizione esegue l'⟨*azione*⟩, verosimilmente la definizione.

Si evitino i comandi nativi di definizione come `\def`, `\edef`, `\gdef` e `\xdef`, di cui si parlerà più avanti nel capitolo 19, perché questi comandi eseguono le definizioni in modo incondizionato e si rischia di azzoppare il sistema. Inoltre si vedrà che nei file di classe e di stile questi comandi sono usati spessissimo; li si lasci dove sono ma non li si usino direttamente; piuttosto se si devono definire comandi ad argomenti delimitati[4] si usi il pacchetto *xparse*, che ricorre a LATEX 3, dopo averne letto e compreso fino in fondo la documentazione.

Se si usa *xpatch* le cose sono decisamente più semplici nel caso di modifiche puntuali a comandi già esistenti, perché non c'è da definire o ridefinire niente

[4] I comandi di LATEX non consentono di definire comandi ad argomenti delimitati.

ma solo da modificare l'esistente già caricato in memoria durante l'esecuzione della compilazione di un dato documento; tuttavia il procedimento, lo si ripete, è più delicato perché bisogna saper mettere la "pezza" (*patch*) al posto giusto avendo interpretato correttamente quanto scritto da altri.

Per le definizioni di nuovi comandi o per la modifica di comandi esistenti bisogna consultare il capitolo 19. Tuttavia la raccomandazione più importante è quella di procedere lentamente con le 'acrobazie' consentite dal linguaggio di programmazione interpretabile dal programma di composizione. Bisogna documentarsi adeguatamente e procedere per gradi; imparare il linguaggio dei programmi di composizione è alla portata di tutti, ma richiede pazienza, attenzione, precisione e costanza.

Non si raccomanderà mai abbastanza: prima di mettersi a creare macro difficili, tortuose, inaffidabili, si cerchino le documentazioni dei pacchetti esistenti. A questo scopo è molto utile essersi caricati una distribuzione completa del sistema TEX. Dopo aver esaminato la documentazione si studino i pacchetti esistenti, anche se all'inizio sembrerà estremamente arduo capirci qualche cosa; ci si abitua con il tempo e con la pazienza. Poi se nonostante tutto si desidera produrre delle macro personali diverse da quelle che si trovano negli archivi, oppure veramente nuove, allora si disporrà di tutti i ferri del mestiere per cimentarsi in prima persona e probabilmente le macro prodotte non saranno affatto difficili, tortuose o inaffidabili.

Un amico che già conosca il linguaggio di sicuro è un grande aiuto, ma, come viene scritto in una delle citazioni riportate alla fine di ogni capitolo nel TEXbook[5]:

> If you can't solve a problem,
> you can always look up the answer.
> But please, try first to solve it by yourself;
> then you'll learn more and you'll learn faster.
>
> — Donald E. Knuth, The TEXbook (1983)

La raccomandazione del padre del sistema TEX deve essere recepita con grande attenzione.

[5]Se non riesci a risolvere un problema, puoi sempre andare a leggere la soluzione [puoi sempre chiedere ad un amico]. Ma per favore, prova prima a risolverlo da solo; imparerai di più e più in fretta.

Capitolo 7

LaTeX: testi speciali

I testi speciali sono quelli che devono venire composti in modo diverso dal resto del testo; in particolare sono speciali quei testi da comporre interrompendo il testo principale oppure quelli da scrivere in parti della pagina diverse da quelle dove si troverebbero normalmente.

I testi che interrompono la composizione normale si chiamano *fuori testo* o, in inglese, in *display*. I brani di testo fuori sequenza sono generalmente costituiti dalle note a piè di pagina, da quelle a margine e da quelle alla fine del documento o di una suddivisione importante del documento, come, per esempio, alla fine dei capitoli.

Anche la matematica può essere composta in display, ma di questo si parlerà più avanti.

7.1 Che cosa sono i testi in display

I testi in display sono quelli 'in vetrina', quelli messi in evidenza in qualche modo o per il loro contenuto o per quello che rappresentano; si distinguono:

- le citazioni
- gli elenchi
- le descrizioni
- le liste bibliografiche

e verranno ora descritti una categoria alla volta.

7.2 Le citazioni

LaTeX offre tre ambienti per mettere in evidenza dei testi citati:

- le citazioni brevi,
- le citazioni lunghe,
- le poesie.

7.2.1 Le citazioni brevi

Una citazione di un testo altrui che consista in poche parole può essere eseguita all'interno del testo racchiudendola fra virgolette. Le virgolette si indicano con i segni `‘‘` per le virgolette aperte e con `’’` per le virgolette chiuse. Siccome la tastiera italiana manca del tasto per segnare l'accento grave, bisogna appoggiarsi alle potenzialità dello *shell editor* per inserire questi segni mancanti, oppure bisogna conoscere i codici scrivibili nel file d'entrata mediante apposite combinazioni di tasti. Le virgolette caporali sono più facili da immettere quando si usa la codifica `T1` per il font di composizione del documento (vedi il capitolo 26); basta scrivere `<<` per i caporali aperti e `>>` per i caporali chiusi; si ottiene « ». Solo che in italiano i caporali sono usati meno frequentemente per le citazioni mentre sono usati più frequentemente per i dialoghi.

Con l'opzione *italian* di `babel`, indipendentemente dalla codifica usata per il testo di uscita, si può ricorrere alle funzionalità del carattere attivo `"` e scrivere `"<` per i caporali aperti e `">` per i caporali chiusi. Inoltre si può scrivere `""` per sfruttare le potenzialità del carattere attivo `"` che introduce in questo caso le doppie virgolette alte aperte “ come se si fossero potuti introdurre da tastiera due accenti gravi in sequenza.

A questo proposito merita ricordare che l'opzione *italian* per `babel` può fare diverse cose oltre a comporre in italiano; in particolare definisce attive le doppie virgolette dritte `"`, definisce il comando `\unit` per inserire le unità di misura, e definisce la virgola in matematica come un carattere intelligente che scopre da solo (quasi sempre correttamente) se svolge le funzioni di separatore decimale o di segno di interpunzione. Ma, si badi bene, nessuna di queste funzionalità è disponibile se non la si attiva; per attivarle quando si usa `pdflatex` e `babel` si usano i seguenti comandi:

```
% prima di \begin{document} ma dopo la chiamata di babel
\setISOcompliance % per attivare \unit
\setactivedoublequote % per attivare le doppie virgolette
% dopo \begin{document}
\IntelligentComma % per attivare la virgola intelligente
\NoIntelligentComma % per disattivare la virgola intelligente
```

Se si usano `xelatex` o `lualatex` con *polyglossia* si può attivare solo il segno delle doppie virgolette diritte mediante l'espressione di una opzione:

```
\setmainlanguage[babelshorthands]{italian}
```

Invece con *polyglossia* i comandi `\setISOcompliance`, `\IntelligentComma` e `\NoIntelligentComma` non sono stati implementati. Chissà, forse nel futuro... L'operazione è difficile, non perché la programmazione di quei comandi lo sia, ma perché il modulo `gloss-italian.ld` di *polyglossia* non è gestito dallo stesso curatore del modulo `babel-italian.ld` per *babel*. Poco male per

`\setISOcompliance`, perché la sua funzionalità è gestita perfettamente dal pacchetto *siunitx*. La virgola intelligente può però essere gestita dai poco conosciuti file di estensione *ncccomma* e *icomma*. In matematica la virgola è trattata di default come un segno di interpunzione; ma le norme ISO prescrivono per tutte le lingue, tranne l'inglese, l'uso della virgola come separatore decimale; ne segue che, scrivendo in italiano senza l'uso della virgola intelligente, quando in matematica si scrive un numero fratto appare uno spazio fine fra la virgola e la cifra che la segue. Il modulo *icomma* in matematica tratta sempre la virgola come separatore decimale, e se la si vuole usare come virgola interpuntiva bisogna lasciare uno spazio dopo la virgola. Il modulo *ncccomma* in matematica tratta la virgola come segno di interpunzione, ma la usa come separatore decimale se e solo se è immediatamente seguita da una cifra. Per questi motivi chi scrive preferisce l'approccio di *ncccomma*. Tuttavia nel capitolo 19 verrà usato questo problema della virgola intelligente per sviluppare diversi esempi con prestazioni diverse e possono andare al di là delle funzionalità dell'approccio di *ncccomma*.

Solo la "virgola intelligente" può essere attivata o disattivata; le altre due funzionalità sono disattive per impostazione predefinita, ma una volta attivate restano tali. Tutto ciò serve per evitare conflitti con altri pacchetti. In particolare ci sono diversi altri pacchetti che definiscono `\unit`; se uno di quei pacchetti viene caricato prima di `\begin{document}`, il comando `\unit` definito da *babel* con l'opzione *italian* resta inerte mentre resta attivo quello del pacchetto invocato. Le doppie virgolette attive a quanto pare danno fastidio ad alcuni utenti che fanno uso del pacchetto *xypic*. La virgola intelligente può interferire saltuariamente con il funzionamento di alcune macro matematiche, ma di solito una volta attivata, non dà problemi di sorta; nel caso si verifichino dei problemi la si può disattivare. Non si ripeterà questo lungo inciso, ma si raccomanda di prendere nota che questo comportamento è disponibile dal 2013 e documenti composti in precedenza potrebbero avere bisogno di quei comandi di attivazione per poter essere composti come prima.

A questo proposito va segnalato che le virgolette aperte " e quelle chiuse " sono asimmetriche; in tipografia non bisogna mai usare le doppie virgolette simmetriche, anche se noti word processor lo fanno. Per questo motivo non bisogna mai usare il segno `"` per inserire le virgolette. Anzi con l'opzione per l'italiano di *babel* il segno `"` è attivo e quindi si comporta come una istruzione. Precisamente se esso è seguito da un qualunque carattere alfabetico esso inserisce un punto di possibile cesura e questa possibilità torna molto comoda per indicare dove andare a capo in quelle parole composte dove si vuole evitare la sillabazione fonetica prescritta dalle grammatiche e dalle norme tipografiche, ma si vuole usare la sillabazione etimologica: se si scrive `macro"istruzione` le due parole `macro` e `istruzione` sono trattate separatamente ai fini della sillabazione, ma una eventuale cesura viene esplicitata all'occorrenza anche fra la 'o' e la 'i' se la cesura in fin di riga lo richiede. Allo stesso modo `"/` consente di andare a capo dopo la barra quando si scrive, per esempio, `modulazione"/demodulazione` che viene composto come modulazione/demodulazione; in fin di riga modulazione/ demodulazione.

Tornando alle virgolette, siccome con la tastiera italiana l'inserzione delle virgolette aperte è problematica, se si usa *babel* con l'opzione per l'italiano, basta scrivere `""` per ottenere “.[1]

7.2.2 Le citazioni lunghe

Le citazioni più lunghe, anche consistenti in qualche riga di testo ma complessivamente senza superare la lunghezza di un capoverso, possono essere evidenziate mediante l'ambiente *quote* in questo modo:

```
\begin{quote}
⟨breve testo da citare⟩
\end{quote}
```

Il testo viene composto mettendolo in evidenza fra margini ristretti rispetto al testo normale.

Per le citazioni più lunghe si usa l'ambiente *quotation*; la sintassi è:

```
\begin{quotation}
⟨testo lungo da citare⟩
\end{quotation}
```

La differenza rispetto all'ambiente *quote* è che in questo caso la citazione può contenere diversi capoversi; questi a loro volta, oltre ad essere composti fra margini ristretti, vengono composti con la prima riga rientrata con il solito rientro di capoverso.

7.2.3 I versi

I versi sono una forma particolare di citazione, nel senso che viene composto un testo scritto da altri; tuttavia si tratta di un testo strutturato in versi e strofe; l'ambiente *verse* richiede che alla fine di ogni verso sia indicato l'ordine di andare a capo mediante il comando `\\` e che ogni strofa sia trattata come un capoverso, cioè che venga lasciata una riga bianca fra una strofa e la successiva.

La sintassi è la seguente:

[1] In generale sarebbe preferibile non usare usare mai la tastiera italiana, ma sempre una tastiera con il layout dei tasti corrispondente alla versione USA e poi usare le comode personalizzazioni per scrivere le lettere accentate, mancanti dalla tastiera USA. La stessa tecnica serve anche per le lettere maiuscole (“È arrivato” invece di “E' arrivato”) e per inserire gli accenti che non si usano in italiano, ma sono necessari per scrivere anche solo poche parole in altre lingue. Peccato che tastiere diverse da quella italiana o portatili dotati di tastiere diverse sono difficilissimi da acquistare nei negozi italiani ma è necessario acquistarle online.

```
\begin{verse}
⟨verso⟩\\
⟨verso⟩\\
...
⟨verso⟩

⟨verso⟩\\
⟨verso⟩\\
...
⟨verso⟩
\end{verse}
```

Ecco, per esempio, il famoso sonetto di Dante Alighieri[2]:

> Tanto gentil e tanto onesta pare
> la donna mia quand'ella altrui saluta,
> ch'ogne lingua deven tremando muta,
> e li occhi no l'ardiscon di guardare.
>
> Ella si va, sentendosi laudare,
> benignamente d'umilta' vestuta;
> e par che sia una cosa venuta
> da cielo in terra a miracol mostrare.
>
> Mostrasi si' piacente a chi la mira,
> che da' per li occhi una dolcezza al core,
> che 'ntender non la puo' chi no la prova;
>
> e par che de la sua labbia si mova
> uno spirito soave pien d'amore,
> che va dicendo a l'anima: Sospira.

viene composto con i comandi seguenti:

```
\begin{verse}
Tanto gentil e tanto onesta pare\\
la donna mia quand'ella altrui saluta,\\
ch'ogne lingua deven tremando muta,\\
e li occhi no l'ardiscon di guardare.

Ella si va, sentendosi laudare,\\
benignamente d'umilta' vestuta;\\
e par che sia una cosa venuta\\
da cielo in terra a miracol mostrare.
```

[2]La mancanza di accenti, sostituiti con gli apostrofi di troncamento, sembra che sia una caratteristica dei manoscritti più accreditati. Probabilmente quegli apostrofi marcano la caduta di una o più lettere finali (apocope) rispetto alla versione latineggiante: umilta' al posto di umiltade; si' al posto di sic, puo' al posto di puote.

```
Mostrasi si' piacente a chi la mira,\\
che da' per li occhi una dolcezza al core,\\
che 'ntender non la puo' chi no la prova;

e par che de la sua labbia si mova\\
uno spirito soave pien d'amore,\\
che va dicendo a l'anima: Sospira.
\end{verse}
```

7.2.4 Brani in linguaggi speciali

Un tipo di citazione o di messa in evidenza di un testo è quello che serve per presentare brani di programmazione, come succede spesso in questo libro. Questo tipo di composizione richiede che nessun carattere sia speciale, che il font usato sia tale da rendere chiaro il fatto che il testo citato è un brano di programmazione, dove gli spazi e le rientranze possono avere un'importanza particolare, dove le linee troppo lunghe vanno a capo ma senza dividere in sillabe nessuna parola e dove anche gli 'a capo' devono essere rispettati senza preoccuparsi di giustificare il testo. Questo risultato si ottiene ricorrendo all'ambiente *verbatim*[3]. L'unica cosa che non si può riprodurre è proprio costituito dalla stringa di caratteri `\end{verbatim}` perché questa stringa, nella sua interezza serve per riconoscere la fine dell'ambiente. Questo testo abbonda di esempi di brani di programmazione; il più recente è quello usato per rappresentare il codice della poesia citata poco sopra. La sintassi dell'ambiente *verbatim* è la seguente:

```
\begin{verbatim}
⟨testo da riprodurre verbatim⟩
\end{verbatim}
```

7.3 Gli elenchi

Gli elenchi sono generalmente di tre tipi: (*a*) in linea con il testo, come questo elenco; (*b*) in display con gli oggetti elencati distinti per mezzo di una numerazione, per cui questo tipo di elencazione si chiama *enumerazione*; e (*c*) in display dove però ogni elemento è evidenziato mediante un simbolo grafico, ma questo simbolo non è distintivo del particolare oggetto elencato; per cui questa è una semplice *elencazione*.

7.3.1 Le elencazioni in linea

Le elencazioni in linea sono quasi sempre composte come l'elencazione precedente, dove il segno distintivo di ogni oggetto elencato è una lettera in corsivo racchiusa

[3] La parola latina 'verbatim' significa 'alla lettera'.

fra parentesi tonde. L'elenco non è particolarmente evidenziato e la presenza delle lettere corsive è l'unico elemento che fa capire che si tratta di una elencazione; non è frequente, ma queste lettere corsive possono essere usate per richiamare l'oggetto identificato, nel qual caso devono essere riportate anche le parentesi, come quando ci si vuol riferire al caso (*a*).

7.3.2 Le enumerazioni

Gli elenchi numerati sono molto più strutturati anche in relazione alla possibilità di annidare un elenco dentro un altro. Le enumerazioni vengono realizzate ricorrendo all'ambiente *enumerate* con la seguente sintassi:

```
\begin{enumerate}
\item[⟨opzione⟩] ⟨oggetto testuale⟩
\item[⟨opzione⟩] ⟨oggetto testuale⟩
...
\end{enumerate}
```

Perciò quella che segue è una enumerazione, dove ogni voce costituisce almeno un periodo completo, cioè comincia con l'iniziale maiuscola contiene almeno un verbo e finisce con il punto fermo.

1. L'enumerazione viene composta dentro un ambiente *enumerate* come era facile prevedere.

2. Ogni oggetto da enumerare viene introdotto con il comando `\item`.

3. Ogni oggetto di una enumerazione può contenere un'altra enumerazione o una semplice elencazione.

4. Se per questo comando non si specifica nessuna ⟨*opzione*⟩ (e quindi non si scrivono nemmeno le parentesi quadre) l'oggetto viene numerato con un 'numero' progressivo.

5. Questo 'numero' può essere qualsiasi oggetto appartenente ad una sequenza ordinata di oggetti, tipicamente:

   - dei numeri arabi, o
   - dei numeri romani, o
   - delle lettere dell'alfabeto latino minuscolo, o
   - delle lettere dell'alfabeto latino maiuscolo.

   La classe del documento specifica quale tipo di numerazione viene eseguito per ogni livello di enumerazione annidata. Di default LaTeX con la sua classe *book* usa i numeri arabi, le lettere minuscole, i numeri romani, le lettere maiuscole.

6. Se invece viene espressa l'opzione, foss'anche una opzione 'nulla', allora invece del 'numero' viene scritta l'opzione.

⋆ Perciò si può fare riferimento simbolico agli elementi numerati per davvero, ma non si può fare riferimento, per esempio a questo elemento, perché non è numerato.

7. Come tutte le liste, le enumerazioni possono essere annidate una dentro l'altra, ma non si possono usare più di tre elencazioni annidate una dentro l'altra dentro una elencazione principale, non tanto perché la cosa sia tecnicamente impossibile, ma perché la struttura diventerebbe troppo complicata e il lettore ne rimarrebbe confuso.

7.3.3 Le elencazioni semplici

Le elencazioni semplici sono in display come le enumerazioni, ma ogni elemento elencato è solamente contrassegnato con un simbolo grafico uguale per tutti gli elementi dello stesso livello di elencazione. L'ambiente si chiama *itemize* e vale la sintassi seguente:

```
\begin{itemize}
\item[⟨opzione⟩] ⟨oggetto testuale⟩
\item[⟨opzione⟩] ⟨oggetto testuale⟩
...
\end{itemize}
```

Perciò:

- il primo livello di elencazione viene contrassegnato con dei pallini neri, ma

⋆ se viene espressa una opzione, come in questo caso, il contrassegno è quanto viene specificato come opzione;

- le elencazioni semplici possono essere annidate l'una dentro l'altra e il contrassegno cambia secondo il livello di annidamento. per esempio:

  – in questo secondo livello il contrassegno è costituito da un tratto medio;

  – si possono cambiare questi contrassegni agendo sulle definizioni che sono contenute nel file di classe;

  vedi anche l'elencazione annidata in una enumerazione nell'elemento 5 della pagina 183;

- in ogni caso si ricorre a queste elencazioni semplici quando gli oggetti da elencare non sono logicamente sequenziali;

- ci si ricordi anche che una elencazione semplice, anche se è in display, fa parte del capoverso che la precede; quindi viene introdotta con una locuzione che termina con i due punti; ogni elemento, in quanto parte di un un unico capoverso, non termina col punto fermo, ma con una virgola o un punto e virgola oppure con una congiunzione; l'ultimo elemento elencato generalmente termina col punto fermo a meno che il capoverso non prosegua dopo l'elencazione.

7.3.4 Alcune osservazioni relative alle elencazioni

Nei paragrafi precedenti si sono descritte alcune elencazioni; la domanda ricorrente è la seguente: Quale tipo di punteggiatura viene usato fra un elemento e il successivo?

Per le elencazioni in linea si suppone che gli oggetti elencati siano brevi frasi, costituite al massimo da un solo periodo, quindi il punto e virgola sembra essere il segno di interpunzione più appropriato. Se non fosse così, allora sarebbe meglio costruire una elencazione in display.

Per le liste in display, semplici o numerate, la punteggiatura può ancora essere un punto e virgola se gli oggetti sono brevi frasi, o frazioni di frase; ma se sono costituiti da periodi completi o da più periodi è necessario usare la punteggiatura che si userebbe anche senza evidenziare l'elencazione. Non necessariamente ogni oggetto elencato deve finire con la stessa punteggiatura, anzi, nelle elencazioni precedentemente usate a titolo di esempio si sono terminati alcuni oggetti senza punteggiatura ma con una congiunzione. Nelle enumerazioni ogni voce è formata da uno o più capoversi completi, quindi il problema non si pone: tutti i capoversi iniziano con una maiuscola e terminano con un punto e a capo.

Nelle enumerazioni è possibile inserire il comando `\label` dentro gli oggetti a cui si vuole fare riferimento in modo simbolico, per esempio si può inserire nell'ultimo elemento dell'enumerazione usata come esempio:

```
\item \label{ele:annidato}Come tutte le elencazioni,
```

così che qui vi si possa fare riferimento mediante l'uso dei comandi `\ref` e `\pageref`, scrivendo

```
l'elemento~\ref{ele:annidato} nella pagina~\pageref{ele:annidato}.
```

ottenendo: "l'elemento 7 nella pagina 184."

7.4 Le descrizioni

Le descrizioni possono essere viste come elencazioni particolari nelle quali il segno distintivo non è né un numero né un simbolo, ma è una parola o una locuzione di cui si fornisce una descrizione o una definizione o una spiegazione nel resto del testo dell'oggetto; la sintassi dell'ambiente *description* è la seguente:

```
\begin{description}
\item[⟨locuzione⟩] ⟨descrizione⟩
\item[⟨locuzione⟩] ⟨descrizione⟩
...
\end{description}
```

Benché la ⟨*locuzione*⟩ sembri facoltativa, essa in effetti è logicamente necessaria, perché non avrebbe nessun senso dare una ⟨*descrizione*⟩ di... nulla.

Questo ambiente è particolarmente indicato per dare una serie di definizioni, oppure per scrivere un glossario.

Esercizio 7.1 Continuando l'esercizio 5.1 si cominci un nuovo capitolo senza specificare l'asterisco, ma dandogli un titolo a piacere; si componga poi un ambiente *description* nel quale si danno le definizioni di scartamento, di corpo e di interlinea.

7.5 Le liste bibliografiche

Le liste bibliografiche sono dei particolari elenchi dove i riferimenti bibliografici sono etichettati con numeri o stringhe che possono essere richiamati simbolicamente nel corso del testo. Va subito detto che qui si parla dell'elenco bibliografico standard, cioè quello che si ottiene utilizzando l'ambiente *thebibliography* con la sintassi seguente per comporre l'elenco dei riferimenti bibliografici:

```
\begin{thebibliography}{⟨stringa⟩}
\bibitem[⟨richiamo⟩]{⟨chiave⟩} ⟨riferimento bibliografico⟩
\bibitem[⟨richiamo⟩]{⟨chiave⟩} ⟨riferimento bibliografico⟩
...
\end{thebibliography}
```

dove

⟨*stringa*⟩ è una stringa di caratteri, generalmente una stringa di cifre, ma il significato numerico è irrilevante; serve all'ambiente per prendere le misure per la lunghezza del campo da lasciare sul margine sinistro dell'elenco al fine di incolonnare correttamente tutte le voci della bibliografia.

⟨*richiamo*⟩ è una parola o una stringa alfanumerica che permette di identificare il particolare riferimento bibliografico all'interno del testo, oltre che nel campo identificativo dell'elenco; se non si specifica niente, (nemmeno le parentesi quadre), il riferimento di default è costituito da un numero, progressivo all'interno dell'elenco, racchiuso fra parentesi quadre. Questo è il modo di citazione più frequente negli articoli scientifici che si riferiscono a scienze nelle quali si fa un grande uso del linguaggio matematico.

Nelle discipline umanistiche è più frequente il modo di citazione 'autore-anno', per cui il ⟨*richiamo*⟩ è appunto costituito dal cognome del (primo) autore seguito dall'anno di pubblicazione dell'opera citata e, in caso di ambiguità, da una lettera progressiva. Ecco allora che la possibilità di stabilire con precisione il nome dell'autore e l'anno di pubblicazione dell'opera citata diventa un ottimo modo per specificare il ⟨*richiamo*⟩.

⟨*chiave*⟩ è il nome simbolico da usare sia nell'eventuale database bibliografico, sia come argomento obbligatorio del comando `\bibitem`, sia come argomento del comando `\cite`, che serve per citare il lavoro all'interno del documento.

L'ambiente *thebibliography*, nella classe `book`, provvede a iniziare una nuova pagina, a scrivere una intestazione di capitolo non numerato, ma intitolata Bibliography, oppure Bibliografia, oppure..., a seconda della lingua di default.

Se questo ambiente contenesse una voce del genere:

```
\bibitem{Lamp94} \textsc{Leslie Lamport},
\textit{A document preparation system --- \LaTeX\ ---
User's guide and reference manual}, Addison Wesley,
Reading, Mass., 2nd ed., 1994.
```

nella bibliografia apparirebbe una voce:

[12] Leslie Lamport, *A document preparation system — LaTeX — User's guide and reference manual*, Addison Wesley, Reading, Mass., 2nd ed., 1994.

e l'opera citata verrebbe citata con `\cite{Lamp94}`, per ottenere nel testo '[12]'.

Se invece nell'elenco bibliografico si fosse scritto:

```
\bibitem[\textsc{Lamport-94}]{Lamp94} \textsc{Leslie Lamport},
\textit{A document preparation system --- \LaTeX\ ---
User's guide and reference manual}, Addison Wesley,
Reading, Mass., 2nd ed., 1994.
```

nella bibliografia apparirebbe una voce

[Lamport-94] Leslie Lamport, *A document preparation system — LaTeX — User's guide and reference manual*, Addison Wesley, Reading, Mass., 2nd ed., 1994.

e l'opera, citata sempre con la stessa chiave `\cite{Lamp94}`, apparirebbe nel testo come '[Lamport-94]'.

Facendo ricorso a pacchetti di estensione da richiamare con `\usepackage` nel preambolo, è possibile comporre questi elenchi in modo stilisticamente diverso; non solo, ma è possibile estrarre da un database bibliografico le informazioni necessarie mediante uno dei programmi BibTeX o biber, così da avere un risultato dalla composizione omogenea oltre che semanticamente completo.

7.6 I riferimenti incrociati

Come si è visto nei paragrafi precedenti, è possibile riferirsi simbolicamente a qualunque oggetto numerato e alla pagina in cui compare, se questo tipo di informazione è sensato.

I comandi per ottenere questi riferimenti sono i seguenti e soddisfano alla sintassi qui di seguito indicata.

```
\label{⟨chiave⟩}
\ref{⟨chiave⟩}
\pageref{⟨chiave⟩}

\bibitem[⟨riferimento⟩]{⟨chiave⟩}
\cite[⟨informazioni aggiuntive⟩]{⟨chiavi⟩}
```

La ⟨*chiave*⟩ è una etichetta simbolica, possibilmente mnemonica (composta con qualunque sequenza di caratteri esclusa la virgola) con la quale si vuole identificare un oggetto. Come si è visto negli esempi precedenti, io compongo la chiave con una breve sigla che ricorda il tipo di oggetto, per esempio `fig:` per una figura, `ese:` per un esempio, `cap:` per un capitolo, `equ:` per una equazione, eccetera, seguito da un nome mnemonico che mi ricordi di che cosa si tratti.

Il comando `\label` con la sua ⟨*chiave*⟩, va inserito nel file sorgente *dopo* il comando che assegna un numero all'oggetto; i comandi `\ref` e `\pageref` con le loro chiavi vanno scritti nel testo nei punti dove si vuole fare riferimento all'oggetto; una precauzione utile dal punto di vista tipografico è quella di far precedere i comandi `\ref` e `\pageref` da una tilde, che in LaTeX ha il significato di *spazio non separabile*, così che durante la composizione il programma non spezzi la riga o non cambi pagina fra il nome che precede il riferimento e il riferimento stesso. Si scriverà, per esempio, `nell'equazione~\ref{equ:Pitagora}` così che TeX non spezzi la riga fra la parola 'equazione' e il numero che esso sostituirà alla ⟨*chiave*⟩ 'equ:Pitagora'.

Facendo uso del pacchetto di estensione *varioref* (da richiamare mediante il comando `\usepackage` nel preambolo) è possibile avere i riferimenti incrociati completi di numero e pagina in locuzioni del tipo 'la figura 5.13 nella pagina seguente' oppure 'la figura 5.14 nella pagina 215' a seconda della distanza del richiamo dal punto in cui l'oggetto compare nel documento.

Per i riferimenti bibliografici il comando `\bibitem`, come si è visto, consente di definire una chiave e, facoltativamente, un richiamo. Il comando `\cite` accetta non solo una chiave, ma diverse chiavi separate da virgole; si potrebbe, per esempio, scrivere `\cite{lamp94,TexComp}` e ritrovarsi scritto nel testo '[12, 13]', cioè la citazione di entrambe le opere identificate mediante i loro numeri progressivi all'interno dell'elenco bibliografico.

Il comando `\cite` accetta anche un argomento facoltativo (che ha senso quando si cita una sola chiave) per indicare un posto preciso dell'opera citata; per esempio, se si scrive `\cite[cap.~4]{Lamp94}` si ottiene '[12, cap. 4]'.

Il vantaggio che questo metodo di citazione simbolico offre è che in caso di correzioni, per esempio cancellazioni o inserimenti di parte del testo, non è necessario correggere tutte le citazioni, ma ci pensa il programma di composizione a correggere tutti i valori che risultano modificati; bisogna solo lanciare il programma un paio di volte, finché nei messaggi finali non compare più il messaggio che 'alcuni riferimenti potrebbero essere stati modificati'.

7.7 Altri testi in display

Talvolta è necessario mettere in evidenza, separandolo dal resto del testo, un brano composto senza giustificazione, o meglio, solo allineato a sinistra, oppure solo allineato a destra, oppure centrato.

Per questo scopo sono disponibili i tre ambienti *flushleft*, *flushright* e *center*.

Si ricorda che il testo giustificato solo a sinistra è più facilmente leggibile di quello giustificato solo a destra e del testo in stile epigrafico con le righe centrate. Tuttavia per i titoli, per esempio, lo stile con le righe centrate è quasi sempre più indicato che non lo stile giustificato solo da un lato o giustificato da entrambi i lati.

Questi tre ambienti compongono il testo che essi racchiudono nella maniera specifica di ciascuno, ma tutti e tre hanno la caratteristica che il testo così composto risulta distanziato sopra e sotto mediante l'equivalente di una riga bianca (vuota).

In certe circostanze non è questo quello che si desidera, ma si vuole cambiare temporaneamente il tipo di giustificazione senza lasciare spazi bianchi prima o dopo il testo così composto.

Esistono i tre comandi (dichiarazioni) `\raggedright`, `\raggedleft` e, naturalmente, `\centering` che ottengono lo scopo desiderato; siccome non esiste un comando `\justify` che permetta di tornare al testo giustificato da entrambi i lati, è necessario limitare l'effetto di questi tre comandi mediante l'uso di un *gruppo*; un gruppo è una parte di testo intercalato a comandi racchiuso: (*a*) fra due parentesi graffe, oppure: (*b*) fra i comandi `\bgroup` ed `\egroup`, oppure: (*c*) fra i comandi `\begingroup` e `\endgroup`, oppure: (*d*) all'interno di ambienti delimitati da `\begin{...}` e `\end{...}`.

Tuttavia bisogna ricordare che il programma di composizione esegue la composizione di ciascun capoverso con la sua giustificazione specifica solo alla fine del capoverso stesso; per indicare esplicitamente la fine di un capoverso all'interno di un gruppo è meglio fare uso del comando `\par` (iniziale di *paragraph*, che in inglese indica il capoverso), che specifica proprio la fine di un capoverso. Perciò scrivere:

```
Andando lungo la strada egli vide una iscrizione
che annunciava:\\[1ex]
{\centering Qui visse Torquato Tasso durante il suo
soggiorno del 1605}\\[1ex] ma la cosa gli sembrava
strana, poiché ricordava che il Tasso era vissuto  nel~'500.
```

è sbagliato perché produce:

> Andando lungo la strada egli vide una iscrizione che annunciava:
>
> Qui visse Torquato Tasso durante il suo soggiorno del 1605
>
> ma la cosa gli sembrava strana, poiché ricordava che il Tasso era vissuto nel ’500.

senza centrare il messaggio lapidario.
Invece scrivendo:

```
Andando lungo la strada egli vide una iscrizione
che annunciava:\par\vspace{1ex}
{\centering Qui visse Torquato Tasso durante il suo
soggiorno del 1605\par}\vspace{1ex}\noindent ma la cosa
gli sembrava strana, poiché ricordava che il
Tasso era vissuto  nel~’500.
```

si ottiene correttamente:

> Andando lungo la strada egli vide una iscrizione che annunciava:
>
> Qui visse Torquato Tasso durante il suo soggiorno del 1605
>
> ma la cosa gli sembrava strana, poiché ricordava che il Tasso era vissuto nel ’500.

Nell’esempio precedente, oltre all’illustrazione dell’uso di `\centering` e del comando `\par`, vengono usati diversi altri comandi non ancora descritti, ma che vale la pena di commentare qui.

`\\` serve per andare a capo sia nel mezzo di un testo, sia quando si compongono matrici matematiche o tabelle; questo comando accetta un argomento facoltativo, racchiuso fra parentesi quadre, che indica quanto spazio verticale lasciare dopo essere andati a capo. `\\[1ex]` vuol dire: ‘vai a capo e lascia uno spazio verticale pari a 1 ex, cioè pari all’altezza di una lettera ‘x’ nel font corrente’. Esiste anche la variante `\\*`, che continua ad accettare l’argomento facoltativo, dove l’asterisco impone a TeX di non andare a pagina nuova con la nuova riga.

Vale la pena di sottolineare che `\\` serve per andare a capo alterando la giustificazione di quella riga all’interno di un capoverso; è errato lasciare una riga vuota e cominciare un altro capoverso con `\noindent`, che solo ad un occhio distratto sembra produrre lo stesso effetto; è altrettanto errato, al contrario, cominciare un altro capoverso scrivendo `\\\indent`, perché si produce la rientranza (*indent* in inglese) ma non si comincia affatto un nuovo capoverso. Con certi stili grafici e/o certi ambienti la rientranza potrebbe essere nulla; con certi altri stili grafici e/o ambienti

la separazione fra i capoversi non è indicata dalla rientranza, ma da una visibile interlineatura.

`\vspace` serve per inserire uno spazio verticale dopo la fine di un capoverso o, detto in termini più tecnici, quando il compositore sta lavorando in modo verticale. L'argomento indica quanto spazio lasciare; siccome questo spazio viene perso all'inizio di una pagina, la forma asteriscata `\vspace*` consente di mantenere lo spazio specificato anche all'inizio della pagina.

`\noindent` serve per evitare il rientro all'inizio di un capoverso.

Attenzione! Sarebbe desiderabile non fare mai uso dei comandi appena descritti; questi non sono parte del mark-up, ma del disegno grafico; sono comandi prescrittivi, non comandi descrittivi. L'autore/compositore dovrebbe attenersi al mark-up descrittivo la maggior parte del tempo. È costretto a ricorrere a questi 'sotterfugi' quando la situazione non prevede qualche ambiente che gli consenta di ottenere la composizione nella maniera desiderata. Piuttosto egli farebbe meglio a definire un nuovo ambiente con le caratteristiche desiderate, ma questo è un argomento che verrà visto molto più avanti.

Nota bene! Al contrario è opportuno usare `\centering` all'interno degli ambienti *figure* e *table* per centrare l'inclusione dell'illustrazione o della tabella da flottare; non occorre servirsi di `\par` perché quegli ambienti contengono già tutte le informazioni per poter gestire questa situazione. Se si usasse invece l'ambiente *center* per centrare l'illustrazione o la tabella, questo ambiente inserirebbe dell'ulteriore spazio bianco prima e dopo la sua apertura e la sua chiusura, per cui l'oggetto flottante risulterebbe poi preceduto e seguito da troppo spazio bianco e/o la didascalia risulterebbe troppo spaziata.

Infatti l'uso di `\centering` per specificare la collocazione orizzontale delle figure e delle tabelle all'interno dei loro rispettivi ambienti flottanti consente di evitare l'inserimento di spazi bianchi in eccesso; per l'uso dettagliato di questi ambienti flottanti si rimanda al paragrafo 9.2; qui li si simulerà per mostrare la differenza fra il risultato ottenibile con il comando `\centering` rispetto a quello ottenibile con l'ambiente *center*; in entrambi i casi vengono collocati due filetti orizzontali per marcare la differente spaziatura verticale.

Come si vede nella figura 7.1, l'oggetto (rappresentato in questo esempio da un semplice rettangolo) viene inserito fra il testo precedente e il testo seguente, insieme alla sua didascalia, distanziato dagli spazi previsti dal file di classe quando si usa `\centering`. Quando invece si usa l'ambiente *center*, questo inserisce ulteriori spazi e l'oggetto e la sua didascalia risultano contornati da troppo spazio bianco.

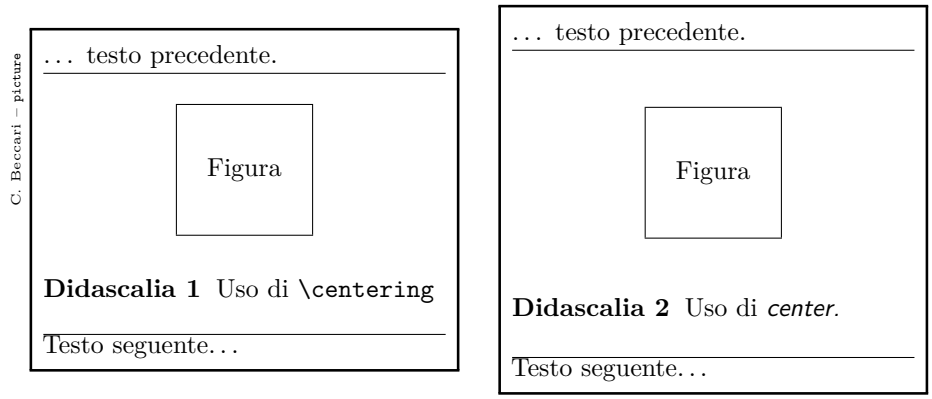


Figura 7.1: Effetto dell'uso del comando `\centering` rispetto all'ambiente *center*.

7.8 Le note

LaTeX consente di scrivere due tipi di note, quelle a piè di pagina e quelle composte nel margine della pagina. Per scrivere note raccolte assieme alla fine di un capitolo, di un articolo, di un libro, bisogna ricorrere al pacchetto esterno *endnotes*[4].

7.8.1 Le note in calce

Per scrivere una nota in calce basta usare il comando `\footnote` con la seguente sintassi:

`\footnote[`⟨*richiamo*⟩`]{`⟨*testo*⟩`}`

Il ⟨*richiamo*⟩ di default è un numero collocato in posizione di apice; talvolta può essere un simbolo come un asterisco, una spada, una doppia spada, eccetera; la numerazione delle note normalmente ricomincia da 1 con l'inizio di un nuovo capitolo, ma la cosa dipende dalla classe del documento. Il ⟨*testo*⟩ della nota non richiede commenti, salvo che si tratta di un 'argomento mobile', e quindi certi comandi 'fragili' non possono essere usati all'interno del ⟨*testo*⟩. Il lettore non si preoccupi dei comandi fragili e degli argomenti mobili; sono cose abbastanza rare quando si scrive un documento 'normale'; in questo testo, che descrive i comandi di LaTeX alcune cose non sono 'normali', proprio perché si tratta di comandi; perciò quando alcuni comandi comparivano nelle note si è dovuto ricorrere a trucchetti vari per poterli scrivere alla lettera, senza farli eseguire.

Le note sono testi speciali, perché vengono composte e trattenute in memoria fino a quando non viene composta la pagina nella quale esse devono comparire;

[4]Non si confonda il pacchetto *endnotes*, formato dal file `endnotes.sty`, con il file `endnote.tex`, che è un file di estensione per Plain TeX, non per LaTeX.

il programma di composizione fa il possibile affinché le note non siano spezzate fra pagine successive e che ognuna sia collocata al piede della pagina nella quale compare il richiamo. È abbastanza raro che esso non riesca a sistemare la nota in calce senza spezzarla, ma può succedere. Non è colpa del programma, ma del testo nel quale la nota è richiamata e del testo della nota. Talvolta, cambiando l'uno o l'altro il problema si risolve da solo.

È possibile attribuire una etichetta ad una nota mediante il comando `\label` inserito insieme alla sua ⟨*etichetta*⟩ alla fine del ⟨*testo*⟩; e a questa etichetta si può fare riferimento per citare la stessa nota in punti diversi del testo.

All'occorrenza si può fare uso di un comando separato per inserire il richiamo e per scrivere il testo; i comandi seguono la sintassi seguente:

```
\footnotemark[⟨richiamo⟩]
\footnotetext[⟨richiamo⟩]{⟨testo⟩}
```

nei quali il ⟨*richiamo*⟩ è facoltativo; se si fa uso dell'argomento facoltativo sia in questi comandi sia in `\footnote` non viene incrementato il contatore delle note e quindi l'attribuzione di una etichetta mediante `\label` non funziona.

Questi due comandi sono prescrittivi, non descrittivi, perché scendono nel dettaglio di come fare apparire le note nel testo composto. Questi comandi, anche se in generale è bene che siano evitati, tuttavia possono essere necessari quando il procedimento normale di composizione non è adeguato al compito. Per esempio non è possibile mettere note dentro alle tabelle; esistono pacchetti di estensione per superare questi limiti, ma se non capita sovente di inserire note nelle tabelle, può valere la pena di usare `\footnotemark` specificando il richiamo dentro una casella della tabella e di usare `\footnotetext` con lo stesso richiamo dopo il comando `\caption` per comporre la didascalia.

Pertanto le note da attribuire alle tabelle vanno composte come mostrato nel modello della pagina 194. Ci si ricordi di specificare una ⟨*larghezza*⟩ adatta a contenere la tabella. Nello stesso tempo non si perda di vista la possibilità di specificare gli argomenti facoltativi sia di `\footnotemark` sia di `\footnotetext`, perché essi devono venire composti con lo stile che LaTeX usa all'interno delle *minipage* (richiami mediante apici letterali in corsivo), ma che, essendo l'ambiente *tabular* composto dentro una scatola, potrebbero non venire stampati correttamente. Esistono ovviamente pacchetti che consentono di eseguire quanto esemplificato qui in modo 'automatico', ma anche Lamport consiglia nel suo manuale di ricorrere all'uso di `\footnotemark` e `\footnotetext` proprio in questi casi.

7.8.2 Le note marginali

Se i margini del documento sono abbastanza ampi, questi possono accogliere le note a margine, talvolta chiamate in latino *marginalia*. Il comando per produrle è `\marginpar` e segue la sintassi:

```
\begin{table}
\begin{minipage}{⟨larghezza⟩}
\centering
\begin{tabular}{...}
... & ... & ... \\
...
... & ... & ...\footnotemark \\
...
\end{tabular}
\caption{⟨didascalia⟩}\label{⟨chiave⟩}
\footnotetext{⟨testo della nota⟩}
\end{minipage}
\end{table}
```

```
\marginpar[⟨nota di sinistra⟩]{⟨nota di destra⟩}
```

Anche le note marginali sono testi che vengono trattenuti in memoria finché la pagina in cui compaiono non viene completamente composta; a seconda del tipo di documento, della classe, dello stile grafico, dell'ampiezza dei margini, la nota marginale può venire collocata nel margine esterno o in quello interno; se però viene collocata a destra del testo, viene composto il testo della ⟨*nota di destra*⟩, altrimenti il testo della ⟨*nota di sinistra*⟩, se è stato specificato, altrimenti il testo della ⟨*nota di destra*⟩ viene usato anche per la ⟨*nota di sinistra*⟩.

Si osservi che se si compone a due colonne, le note relative alla colonna di sinistra vengono collocate a sinistra della colonna e, corrispondentemente le note relative alla colonna di destra vengono collocate a destra della colonna. Componendo ad una colonna le note vengono collocate sempre nel margine esterno, quindi a destra del testo nelle pagine di destra e a sinistra del testo nelle pagine di sinistra. Sembra complicato, ma a pensarci bene è del tutto logico. È vero che con l'apposito comando `\reversemarginpar` questo comportamento può venire scambiato, ma in generale si suppone che questa scelta sia fatta nel file di classe e il compositore non debba preoccuparsene.

Le note marginali vengono composte in colonne strette, con una giustezza che raramente supera i 40 mm; con una giustezza così piccola è possibile che la composizione a pacchetto (giustificata da entrambi i lati) produca una composizione con ampi spazi bianchi fra le parole e con parole difficili da dividere in sillabe che sporgono fuori dalla giustezza. In questi casi sarebbe meglio specificare per la nota di destra una composizione giustificata solo a sinistra; basta usare il comando `\raggedright`. Per le note di sinistra, per una questione di simmetria, sarebbe quindi desiderabile specificare un testo giustificato solo a destra mediante il comando `\raggedleft`. La cosa è evidentemente possibile se si fa uso dell'argomento facoltativo descritto nella sintassi di `\marginpar`. Tuttavia la lettura di un testo giustificato solo a destra può risultare meno agevole di un testo giustificato solo a sinistra; ecco perché, pur essendo possibile, raramente si

specifica una composizione giustificata solo a destra, anche se i testi delle note marginali sono solitamente molto brevi e quindi non sono faticosi da leggere anche se fossero giustificati solo a destra.

Le note marginali sono collocate nel margine che compete loro mantenendo la loro prima riga allineata con la riga del testo alla quale esse si riferiscono; se due note marginali compaiono troppo ravvicinate e ci fosse il rischio che si sovrappongano, allora LaTeX provvede a spostare in basso le note che potrebbero interferire con le note precedenti assicurando uno spazio di separazione specificato nel file di classe; come si vede nelle due note qui a margine, la seconda, specificata nella riga dove si dice "compaiono troppo ravvicinate...", risulta troppo ravvicinata alla prima ed è stata leggermente abbassata. Se una nota marginale viene associata ad una delle ultime righe della pagina, essa può fuoriuscire dalla gabbia di composizione in modo non accettabile; bisogna provvedere quindi a modificare il testo da annotare o a spostare manualmente la nota marginale. Si capisce, perciò, perché è opportuno che il testo delle note marginali sia brevissimo e che possibilmente non superi la lunghezza di una riga.

Questa è una nota marginale che si sviluppa su più righe.

Questa è una seconda nota marginale troppo vicina alla nota precedente.

Le note marginali vengono usate spesso negli scritti letterari, specialmente se si tratta di note brevi e frequenti; è raro incontrare note marginali in scritti tecnici.

7.9 Un esempio specifico di testi speciali

Scrivendo libri didattici capita di inserire degli esercizi; direi anzi che è buona norma inserire molti esercizi, alcuni svolti, altri da svolgere; alcuni con le soluzioni che li seguono direttamente, altri che hanno le soluzioni in un'apposita appendice, altri ancora senza soluzione. Nei testi statunitensi, spesso vengono fornite in appendice solo le soluzioni degli esercizi numerati con un numero dispari, lasciando gli esercizi pari agli studenti, affinché imparino a lavorare in modo autonomo e imparino anche ad analizzare la soluzione per verificarne la correttezza.

Gli esercizi spesso dopo aver presentato una panoramica del problema da risolvere, propongono diverse domande, spesso strutturate gerarchicamente; le soluzioni, se fornite, devono rispettare lo stesso schema. Riconosciamo subito che queste domande gerarchiche formano tipicamente una lista strutturata, quindi per questi insiemi di domande e risposte gli ambienti delle liste descritte nei paragrafi precedenti devono essere usati esplicitamente o implicitamente.

Ma c'è un altro problema: per chi compone il libro è importante che il testo di ciascun esercizio sia immediatamente seguito dalla sua soluzione, cosicché se l'intero esercizio è da sostituire con un altro diventa quasi immediato selezionare l'intero blocco delle righe del testo da sostituire e cancellarlo, sostituendolo con un altro testo dell'esercizio e della sua soluzione. Per la validità didattica, invece, non è mai opportuno che la soluzione segua immediatamente l'enunciato dell'esercizio; infatti solitamente le soluzioni sono raccolte in una apposita appendice.

Certo non sarebbe troppo difficile comporre il testo sorgente delle soluzioni in un file separato e includere questo file verso la fine del testo. Ma ogni volta

che si deve eseguire qualche correzione o sostituzione è facilissimo commettere errori e sostituire uno dei testi (enunciato o soluzione) con testi spaiati. Mi spiego: l'esercizio 3.15 è abbinato alla sua soluzione 3.15; ma se i due testi nei file sorgente sono distanti l'uno dall'altro potrebbe succedere di sostituire l'enunciato 3.15 con un altro enunciato, la cui soluzione, però viene sostituita per errore alla soluzione dell'enunciato 3.16.

È chiaro che le soluzioni e gli enunciati fanno riferimento le une agli altri mediante il meccanismo dei riferimenti incrociati resi con `\ref` e `\label` e le stesse chiavi. Ma è facile sbagliarsi, quindi è meglio ridurre le possibilità di errore mantenendo l'enunciato adiacente alla sua soluzione, magari nello stesso file da sostituire eventualmente con un altro file contenente un altro enunciato con la sua soluzione.

In questo modo le possibilità d'errore sono davvero minime.

Ma nasce il problema: come si fa a dilazionare la stampa della soluzione al punto giusto fra le appendici? La cosa non è facile per niente, non parliamo di eseguirla con un normale word processor. Ma con LaTeX si può fare.

Basta caricare il pacchetto *exercise*; questo pacchetto essenzialmente definisce due ambienti: *Exercise* e *Answer*, il primo per contenere l'enunciato dell'esercizio e il secondo per contenere la sua soluzione. Entrambi richiedono di inserire delle opzioni del tipo ⟨*chiave*⟩=⟨*valore*⟩. Perché i due ambienti si riferiscano l'uno all'altro, fra le opzioni di *Exercise* bisogna inserire `label={`⟨*etichetta*⟩`}` e fra le opzioni di *Answer* bisogna inserire `ref={`⟨*etichetta*⟩`}`, usando la stessa ⟨*etichetta*⟩. Naturalmente si possono inserire altre parole chiave che è meglio esaminare studiando attentamente la documentazione del pacchetto *exercise*.

Non si scende in ulteriori dettagli. Il concetto che si vuole evidenziare è che quando si compone il testo, il contenuto dell'ambiente *Exercise* viene composto subito nel file di uscita, mentre il contenuto dell'ambiente *Answer* viene composto dentro un registro della memoria del programma di composizione e ogni successiva soluzione di altri esercizi viene accodata a questo registro. Al momento di stampare le soluzioni, questo registro viene svuotato nel flusso di testo composto destinato al file di uscita. Semplice a dire, ma creare le macro del pacchetto *exercise* ha richiesto non poca immaginazione e competenza al suo autore.

Il seguente banale esercizio illustra quanto si può fare con questo pacchetto; ovviamente si possono specificare molte opzioni in modo da poter personalizzare l'aspetto dell'enunciato e, rispettivamente, l'aspetto della soluzione. si possono aggiungere anche alte informazioni, alcune delle quali rappresentano solo delle annotazioni per il compositore. L'intestazione dell'enunciato e quella della soluzione possono venire personalizzate in molti modi. In ogni caso essi possono essere localizzati in una lingua particolare se (*a*) il pacchetto *exercise* viene caricato dopo *babel* o *polyglossia*, e (*b*) se le definizioni nella lingua principale del documento sono già esistenti nel pacchetto. Nel momento di scrivere questo testo (maggio 2015) il pacchetto contiene le localizzazioni per l'inglese, il francese, lo spagnolo, l'italiano, l'olandese, il tedesco, il portoghese e il russo; l'autore attende dagli utenti le traduzioni per altre lingue, come succede normalmente per il software libero.

Esercizio 1

Calcolare la lunghezza del secondo cateto in un triangolo rettangolo con il primo cateto che vale 12 cm e l'ipotenusa che vale 13 cm.

Il testo sorgente di questo esercizio e della soluzione è il seguente:

```
\begin{Exercise}[label={pitagora},difficulty={0}]
Calcolare la lunghezza del secondo cateto in un triangolo
rettangolo con il primo cateto che vale 12\unit{cm} e
l'ipotenusa che vale 13\unit{cm}.
\end{Exercise}

\begin{Answer}[ref={pitagora}]
Dal teorema di Pitagora sappiamo che:
\[a^2 + b^2 = c^2\]
dove $a$ e $b$ sono i cateti e $c$ è l'ipotenusa si ha:
\begin{align*}
12^2 + b^2 &= 13^2\\
144 + b^2  &= 169\\
\intertext{da cui:}
b^2        &= 169 - 144 = 25\\
b          &= \pm5\\
\intertext{e, avendo eliminato la soluzione negativa, la soluzione
           dell'esercizio è:}
b          &= 5\unit{cm}
\end{align*}
\end{Answer}
```

In questo esempio non si stampa subito la soluzione ma la si compone solo alla fine di questo capitolo mediante il comando `\shipoutAnswer`.

Soluzione dell'esercizio 1

Dal teorema di Pitagora:

$$a^2 + b^2 = c^2$$

dove a e b sono i cateti e c è l'ipotenusa si ha:

$$\begin{aligned} 12^2 + b^2 &= 13^2\\ 144 + b^2 &= 169 \end{aligned}$$

da cui:

$$\begin{aligned} b^2 &= 169 - 144 = 25\\ b &= \pm5 \end{aligned}$$

e, avendo eliminato la soluzione negativa, la soluzione dell'esercizio è:

$$b = 5\,\mathrm{cm}$$

Capitolo 8

LaTeX: tabelle

8.1 Introduzione

Le tabelle sono oggetti strutturati, contenenti testo o formule o espressioni matematiche, formati da un certo numero di celle ordinate in senso orizzontale e in senso verticale; queste celle possono essere riquadrate mediante filetti; spesso ciascuna colonna di celle ha una prima cella il cui contenuto descrive quello delle celle sottostanti.

Comporre questo insieme di celle è sempre stato un grosso problema anche ai tempi in cui la composizione veniva eseguita a mano; ora i vari word processor e text processor consentono di lavorare con più facilità, ma senz'altro la composizione delle tabelle, specialmente delle tabelle ben composte, richiede molta pazienza e molta preparazione iniziale; se il compositore si fa uno schizzo, anche grossolano, della tabella, sa già in anticipo dove sorgeranno i maggiori problemi e quindi sa predisporre gli accorgimenti adeguati.

Qui si parlerà di tabelle 'testuali'; quelle con contenuto prevalentemente matematico si chiamano in generale *matrici* e verranno viste nel capitolo 14.

Distinguiamo subito due casi, cioè quello delle tabelle che non occupano più di una pagina, e le tabelle lunghe, che occupano più pagine. Le tabelle 'brevi', nonostante la loro brevità, sono oggetti ingombranti, perciò è meglio che il programma di composizione sposti la tabella nel documento composto dove c'è sufficiente posto per collocarla, talvolta nella stessa pagina o persino nel punto stesso dove è stata definita, più spesso all'inizio o alla fine della pagina successiva, talvolta alla fine di un capitolo o dell'intero documento.

Per consentire questo trattamento il programma deve conservare in memoria la tabella composta, per poi scaricarla nel file di uscita alla prima occasione buona.

Questo modo di procedere implica due cose:

- le tabelle devono essere numerate per farvi riferimento mediante i loro numeri;

- le tabelle troppo grandi possono bloccare la memoria perché vi vengono trattenute in quanto LaTeX non trova mai l'occasione buona per trasferirle nel file di uscita.

Il fatto che le tabelle siano numerate e che debbano essere fatte 'flottare' fino al punto più idoneo richiede un ambiente specifico di 'flottaggio', e implicano un comando per eseguire la didascalia, anche una semplice didascalia composta dal solo titolino.

8.2 Come far flottare una tabella

L'ambiente di flottaggio si chiama *table* e richiede la seguente sintassi:

```
\begin{table}[⟨codici di posizionamento⟩]
⟨didascalia e tabella vera e propria⟩
\end{table}
```

I ⟨*codici di posizionamento*⟩ sono i codici con i quali viene descritto il modo di collocare la tabella nel testo; di default questi codici sono `h`, `t`, `b` e `p`; i codici sono complessivamente i seguenti:

`h`	poni la tabella qui (*here* in inglese)
`t`	poni la tabella in testa alla pagina (*top*)
`b`	poni la tabella al fondo della pagina (*bottom*)
`p`	poni la tabella in una pagina di soli oggetti flottanti (*page*)
`!`	poni la tabella dove prescritto dagli altri codici di posizione senza essere troppo rigoroso con i parametri di posizionamento

Per maggiori dettagli si veda il paragrafo 9.2 dove sono illustrati vari metodi per controllare la posizione degli oggetti flottanti e dove si descrivono alcuni pacchetti che consentono di modificare gli ambienti standard e di definire nuovi ambienti flottanti.

8.3 Le didascalie

Quanto segue vale per tutte le didascalie, anche per quelle delle figure. Come impostazione predefinita nelle classi standard LaTeX pone la didascalia sotto alle tabelle; in Europa, e in Italia in particolare, viene raccomandato di inserire la didascalia prima della tabella a cui si riferisce. Per risolvere questo problema o si usano classi non standard, come *memoir*, oppure si usa il pacchetto *caption* impostandolo adeguatamente. Invece per le figure si preferisce la didascalia dopo l'illustrazione. Ovviamente questa è una affermazione generale, visto che le didascalie possono trovare diverse posizioni a seconda del tipo di illustrazioni o di tabelle e del tipo di didascalie.

Nuovamente, per usare didascalie laterali o composte con stili diversi da quello preimpostato con le classi standard, o si usano classi non standard o si usa il pacchetto di estensione *caption*.

La didascalia viene composta mediante il comando `\caption` secondo la seguente sintassi:

```
\caption[⟨didascalia breve⟩]{⟨didascalia⟩}\label{⟨etichetta⟩}
```

La ⟨*didascalia*⟩ è il testo a cui è stato premesso il titolino 'Tabella', o 'Table', o 'Tableau', o ..., a seconda della lingua in uso, seguito dal numero progressivo della tabella; questa parte viene inserita di default; il resto della didascalia può anche essere omesso, ma non sarebbe una buona idea. Il resto della didascalia può essere composto con un titolo seguito da un capoverso descrittivo. Questo capoverso può mancare; se c'è, viene terminato con il punto fermo. Se non c'è, il titolo viene terminato senza punto fermo, come tutti i titoli.

Non è il caso che l'intera ⟨*didascalia*⟩ vada a finire nell'elenco delle tabelle, al massimo ci andrà il titolo, ma non certo il capoverso descrittivo. Ecco, quindi, che la ⟨*didascalia breve*⟩ torna utile per essere inserita nell'elenco delle tabelle, eventualmente abbreviando ulteriormente il titolo rispetto a quello inserito nella didascalia completa.

L'⟨*etichetta*⟩ introdotta con il comando `\label` serve per poter fare riferimento al suo numero (assegnato automaticamente da LaTeX) mediante un nome simbolico; è opportuno, quindi, che l'⟨*etichetta*⟩ sia scelta con qualche criterio mnemonico in modo che essa richiami il contenuto della tabella, piuttosto che un'altra caratteristica qualsiasi. Nel corso del testo la tabella viene richiamata per numero mediante l'uso del comando `\ref` e/o per numero di pagina mediante l'uso del comando `\pageref`, secondo la sintassi seguente:

```
\ref{⟨etichetta⟩}
\pageref{⟨etichetta⟩}
```

È opportuno fare uso del segno di legatura `~` per non staccare il riferimento della tabella dal nome che l'accompagna; si scriverà pertanto:

```
... la tabella~\ref{tab:popolazione} mostra che ...
```

per riferirsi alla tabella etichettata con `tab:popolazione` con la certezza che non potrà avvenire un fine riga o un fine pagina fra la parola 'tabella' e il suo riferimento numerico.Vale la stessa raccomandazione per riferirsi al numero di pagina tramite `\pageref`.

Per modificare lo stile compositivo delle didascalie esiste l'ottimo pacchetto di estensione *caption* mediante il quale si possono modificare tutti i possibili parametri che presiedono alla loro composizione.

8.4 Come comporre la tabella vera e propria

La tabella vera e propria viene composta mediante l'ambiente *tabular* che richiede la seguente sintassi:

```
\begin{tabular}[⟨allineamento⟩]{⟨descrittori delle colonne⟩}
⟨prima cella⟩&⟨seconda cella⟩&...\\
⟨prima cella⟩&⟨seconda cella⟩&...\\
...\\
⟨prima cella⟩&⟨seconda cella⟩&...\\
\end{tabular}
```

C'è anche la versione asteriscata *tabular** che obbedisce alla sintassi seguente:

```
\begin{tabular*}{⟨larghezza⟩}[⟨allineamento⟩]{⟨descrittori delle colonne⟩}
⟨prima cella⟩&⟨seconda cella⟩&...\\
⟨prima cella⟩&⟨seconda cella⟩&...\\
...\\
⟨prima cella⟩&⟨seconda cella⟩&...\\
\end{tabular*}
```

Per l'ambiente *tabular** si vedano i dettagli nel paragrafo 8.5; qui, per ora, concentreremo l'attenzione sull'ambiente *tabular*, ma ricordiamo che quasi tutto quello che verrà detto qui vale anche per le tabelle dell'ambiente *tabular**.

8.4.1 I descrittori delle colonne

I descrittori delle colonne sono esposti nella tabella 8.1.

Mentre tutto il resto sembra ovvio, è necessario spiegare che cosa sia una *@-espressione*; questa è una sequenza di comandi, di descrittori, di elementi testuali da mettere fra due celle adiacenti al posto del normale spazio fra le colonne. Anche il descrittore `|` potrebbe essere visto come un separatore fra colonne adiacenti (e lo è) ma non è una @-espressione; questa sostituisce *anche* lo spazio di separazione fra le celle adiacenti, e quindi fra le colonne. Per esempio, si può notare che la tabella 8.1 ha i filetti orizzontali che sporgono un poco fuori dei limiti della cella di sinistra e di destra. Generalmente l'effetto estetico è gradevole, ma potrebbe essere necessario eliminare tale spazio; serve allora inserire come primo e ultimo descrittore di colonna una @-espressione 'vuota': `@{}`.

Il parametro facoltativo di ⟨*allineamento*⟩ serve per allineare una tabella con il testo circostante; oppure, se si mettono due o più tabelle affiancate dentro lo stesso ambiente *table*, conviene allinearle verticalmente in modo adatto a seconda del loro contenuto; i rispettivi codici sono `t` (*top*), `b` (*bottom*) e `c` (*center* – preimpostato); si vedano i dettagli nelle pagine 766 e 766.

Esercizio 8.1 Una @-espressione potrebbe essere usata per ripetere come separatore di colonna uno stesso segno; in una tabella che riportasse una colonna con valori numerici fratti e si volesse incolonnare questi numeri in modo che il separatore decimale sia incolonnato fra tutte le celle, indipendentemente dal numero delle cifre prima e dopo la virgola, si potrebbe usare la virgola come testo all'interno della @-espressione spezzando la colonna in due colonne adiacenti e incolonnare la parte intera in una colonna allineata a destra e la parte decimale in una colonna allineata a sinistra.

Predisporre una tabella con una colonna di testi descrittivi per ogni riga e una 'colonna' di numeri decimali incolonnata sulle rispettive virgole decimali.

Descrittore	Spiegazione
`l`	Cella col contenuto allineato a sinistra
`c`	Cella con il contenuto centrato
`r`	Cella con il contenuto allineato a destra
`p{`⟨*larghezza*⟩`}`	Cella contenente un blocco di testo giustificato e largo ⟨*larghezza*⟩
`|`	Filetto verticale
`@{`⟨*espressione*⟩`}`	@-espressione
`\vline`	Filetto verticale da sostituire a `|` dentro una @-espressione
`*{`⟨*numero*⟩`}{`⟨*descrittori*⟩`}`	Ripetizione di ⟨*numero*⟩ volte la stessa sequenza di ⟨*descrittori*⟩

Tabella 8.1: Descrittori delle colonne per le tabelle

L'esercizio precedente è utile per capire il meccanismo; per fortuna il pacchetto di estensione `array` consente già mediante un nuovo descrittore di incolonnare i numeri fratti sulla base del loro separatore decimale. Le estensioni che questo pacchetto consente sono tali e tante che converrebbe richiamarlo costantemente per ogni documento da comporre. Si rinvia alla documentazione di quel pacchetto per maggiori dettagli.

Il descrittore `p`, come già detto, permette di comporre il contenuto di una cella come un piccolo capoverso giustificato a pacchetto (da entrambi i lati); dato che le celle hanno una larghezza limitata e certamente minore di quella dell'intero testo, talvolta è consigliabile specificare `\raggedright` oppure, meno frequentemente, `\raggedleft`. Ci si ricordi che il comando `\\` serve sia per andare a capo in un capoverso, sia per terminare la riga di una tabella. Bisogna quindi stare attenti, quando si usa il descrittore `p`, per quale scopo si usa il comando `\\`; se serve per andare a capo dentro il capoverso si evita ogni fastidio se si usa al suo posto `\newline`, che non consente opzioni, ma che certamente

Opera	Brano
I promessi sposi	Quel ramo del lago di Como, che volge a mezzogiorno, tra due catene non interrotte di monti, tutto a seni e a golfi, a seconda dello sporgere e del rientrare di quelli, vien quasi a un tratto...

Tabella 8.2: Tabella la cui colonna di destra è specificata con `p{80mm}`

non è ambiguo. Si osservi infine nella tabella 8.2 l'allineamento dei contenuti delle celle quando si usi il descrittore `p`.

8.4.2 Il raggruppamento delle celle

Le tabelle possono avere alcune celle adiacenti raggruppate a formare un'unica cella. LaTeX di per sé consente di raggruppare le celle solo orizzontalmente; mediante il pacchetto di estensione *multirow* è possibile raggrupparle anche verticalmente, ma il pacchetto è, per così dire, ancora in evoluzione, per cui bisogna supplire a certe piccole manchevolezze con un po' di tentativi e di correzioni successive. In generale, però, una tabella ha un aspetto più professionale se non contiene celle raggruppate verticalmente.

Si usa il comando `\multicolumn` con la sintassi seguente per raggruppare alcune celle orizzontalmente:

`\multicolumn{`⟨*numero*⟩`}{`⟨*descrittore*⟩`}{`⟨*contenuto*⟩`}`

Il ⟨*numero*⟩ specifica quante celle bisogna raggruppare; il ⟨*descrittore*⟩ contiene i codici di allineamento e di separazione necessari per descrivere l'unica grande cella frutto del raggruppamento; il ⟨*contenuto*⟩ è il testo contenuto nell'unica grande cella.

Bisogna fare attenzione ad un dettaglio: i separatori di colonna da inserire nel ⟨*descrittore*⟩ riguardano solo il lato *destro* dell'unica cella, a meno che la cella non sia la prima della serie di celle in una riga: in questo unico caso bisogna esplicitare anche il descrittore da inserire nel lato sinistro della cella.

Per esempio, se si avesse una tabella in cui il campo dei descrittori fosse il seguente:

```
\begin{tabular}{|*6{c|}}
```

e si volessero raggruppare le celle 3, 4 e 5 in un'unica grande cella presente in una sola riga (per esempio la riga di intestazione), bisognerebbe specificare nella posizione della cella 3

```
\multicolumn3{c|}{Intestazione 3}
```

mentre se si volessero raggruppare le celle 1 e 2, sarebbe necessario specificare:

```
\multicolumn2{|c|}{Intestazione 2}
```

Per cui la prima riga della tabella, verosimilmente sarebbe descritta dal seguente codice:

```
\multicolum2{|c|}{Intestazione 2}
             &\multicolum3{c|}{Intestazione 3}&Intestazione 1\\
```

Nei precedenti esempi si noti che il numero di celle da raggruppare è formato da una sola cifra quindi non ha bisogno di essere racchiuso fra parentesi graffe; questa è una regola generale; quando un argomento obbligatorio è formato da un solo oggetto (*token* nel gergo di LaTeX) non ha bisogno delle parentesi graffe, le quali, invece, sono necessarie ogni volta che l'argomento è costituito da diversi token; ovviamente non è vietato usare le graffe anche quando l'argomento è costituito da un solo token, ma omettendole si scrive un po' di meno e non si corre il rischio di dimenticare la graffa chiusa.

8.4.3 I separatori verticali

Per separare verticalmente le righe di celle generalmente non serve nulla; le tabelle sono più professionali se non contengono filetti o se ne contengono pochissimi.

In ogni caso LaTeX consente di separare le celle con dei filetti o degli spazi aggiuntivi.

`\\` il comando `\\` serve per terminare una riga di celle; però esso consente anche di specificare uno spazio verticale da aggiungere alla normale spaziatura; si scrive allora, per esempio, `\\[1.5ex]` per terminare la riga e aggiungere uno spazio ulteriore di 1,5 ex fra il fondo della riga corrente e l'inizio della riga seguente. Si faccia attenzione che all'inizio della prima cella di ogni riga di qualunque tabella è inserito un oggetto invisibile che si chiama `\strut` (*pilastrino* in italiano); esso serve per garantire una altezza e una profondità minima ad ogni riga, in modo che le righe sembrino distanziate uniformemente indipendentemente dai loro ascendenti e discendenti; se come argomento facoltativo di `\\` si specifica uno spazio troppo piccolo, questo potrebbe risultare assorbito dalla profondità dello `\strut` della riga corrente.

`\tabularnewline` serve per terminare una riga della colonna e per passare alla riga successiva (vale anche per l'ambiente *array* da usarsi in ambiente matematico); sostanzialmente agisce come `\\`, ma si riferisce esclusivamente all'ultima cella di una riga della tabella o della matrice; esso è utilissimo, per non dire indispensabile, quando l'ultima cella della riga è caratterizzata dal descrittore `p` (oppure con il pacchetto *array* anche con i descrittori `m` e `b`). Infatti in questi casi il comando `\\` è ambiguo: serve per andare a capo in una riga della cella oppure serve per andare a capo in una tabella? Se non si prendono particolari precauzioni il comando `\\` viene interpretato come fine riga del testo della cella, ma usando `\tabularnewline` si toglie ogni ambiguità e la tabella o la matrice viene composta correttamente.

\hline serve per tracciare un filetto orizzontale attraverso tutta la tabella; due comandi **\hline** di seguito l'uno all'altro producono due filetti distanziati di un paio di punti tipografici che rendono maggiormente evidente la separazione verticale. Il comando **\hline** deve essere specificato o prima della prima riga di celle, o dopo un comando ****.

\cline serve per tracciare un filetto orizzontale sotto *alcune* colonne adiacenti che devono venire specificate mediante la sintassi:

```
\cline{⟨colonna iniziale⟩-⟨colonna finale⟩}
```

Si noti che si possono ripetere due comandi **\cline** di seguito specificando le stesse colonne iniziale e finale ma non si ottiene un filetto doppio perché i due filetti si sovrappongono; si possono invece inserire due specificazioni con colonne diverse per mettere due filetti di seguito uno all'altro in orizzontale.

Esercizio 8.2 Si predisponga una tabella a più colonne e si inseriscano contenuti qualsiasi nelle celle; ma si specifichino tutti e tre i tipi di separatori verticali descritti in questo paragrafo, osservando bene che cosa succede quando lo spazio che costituisce l'argomento facoltativo del comando **** è di pochi punti. Ripetere la spaziatura con diversi valori di quello spazio.

8.4.4 Come rendere le tabelle un poco più aperte

Nonostante la presenza di uno **\strut** nella prima cella di ogni riga, talvolta sembra che questo **\strut** non sia abbastanza grande cosicché i filetti orizzontali sembrano toccare le lettere maiuscole o i discendenti delle minuscole.

Si può usare il parametro **\arraystretch** con la seguente sintassi:

```
\renewcommand\arraystretch{⟨fattore⟩}
```

Il ⟨*fattore*⟩ è un numero decimale, generalmente maggiore di uno, che rappresenta il fattore di scala con cui si vuol ingrandire verticalmente ogni spazio prodotto dagli **\strut**, e quindi con il quale si vuole ingrandire l'intera tabella. Per esempio si può dichiarare:

```
\begin{table}
\renewcommand\arraystretch{1.1}
\begin{tabular}{...}
...
\end{tabular}
\end{table}
```

In questo modo il valore unitario di default del parametro **\arraystretch** viene aumentato del 10% e tutte le righe della tabella risultano del 10% più alte e più profonde. Il fatto di cambiare il valore di **\arraystretch** all'interno di

un ambiente, *table* in questo caso, confina la modifica all'interno dell'ambiente stesso.

Attenzione, così facendo si ingrandiscono verticalmente tutte le righe della tabella, anche quelle che non sono precedute o seguite da filetti orizzontali.

Può darsi che questo sia quello che si desidera, specialmente se si sta usando un fattore di scala non tanto superiore a uno. Tuttavia potrebbe essere una scelta non conveniente. Quando si vogliono ingrandire solo le celle precedute e/o seguite da filetti orizzontali non resta che inserire direttamente un altro `\strut` nelle righe che interessano. Il pilastrino che il compositore può usare si ottiene con il comando `\rule` e la sua sintassi è la seguente:

`\rule[`⟨*innalzamento*⟩`]{`⟨*base*⟩`}{`⟨*altezza*⟩`}`

Questo `\rule` definisce un riga verticale larga ⟨*base*⟩ e alta ⟨*altezza*⟩, innalzata rispetto alla linea di base del testo di ⟨*innalzamento*⟩. Per rendere questa riga invisibile basta specificarne la base nulla; per l'altezza si specifica quello che si vuole e si può innalzare o abbassare (innalzamento negativo) questa riga di quanto si vuole.

Personalmente io uso queste definizioni:

```
\begin{table}
\def\U{\rule{0pt}{3ex}}
\def\D{\rule[-0.5ex]{0pt}{0pt}}
\def\V{\U\D}
\begin{tabular}{...}
...
\end{tabular}
\caption{...}\label{...}
\end{table}
```

e inserisco `\U` nella prima cella di una riga *preceduta* da un filetto; inserisco `\D` nella prima cella di una riga *seguita* da un filetto; inserisco `\V` nella prima cella di una riga *preceduta e seguita* da un filetto.

Il comando `\def` appartiene al linguaggio base di TEX e sarebbe meglio non usarlo mai quando si scrive con LATEX. Infatti questo comando definisce una nuova istruzione e, per sbaglio, si potrebbe ridefinire una istruzione interna di LATEX perché non viene eseguita nessuna verifica della liceità di questa definizione. Tuttavia in questo caso le tre istruzioni `\U`, `\D` e `\V` non corrispondono a nessun'altra definizione interna a LATEX, ma, quello che più conta, la definizione mediante `\def` rimane valida solo all'interno dell'ambiente *table*, per cui all'uscita dall'ambiente, dopo `\end{table}`, quei comandi riprendono il significato che avevano prima di entrare nell'ambiente *table*.

Così ho fatto in molte tabelle esposte finora e così farò alcune delle prossime tabelle. Così facevo con tutte le tabelle quando ho composto le prime versioni di questo libro.

Esiste però la possibilità di usare il pacchetto `booktabs`; questo contiene le definizioni per i tre filetti `\toprule`, `\midrule` e `\bottomrule` da mettere rispettivamente sopra la riga di intestazione, sotto la riga di intestazione e sotto l'ultima riga della tabella. Il primo e l'ultimo sono leggermente più spessi del filetto intermedio. Ovviamente tutti i parametri stilistici sono configurabili.

Questi filetti assicurano uno spazio adeguato in modo che risultino sufficientemente distanziati dal contenuto delle celle adiacenti. Sono incompatibili con i filetti verticali, quindi 'obbligano' a comporre le tabelle in modo professionale, perciò questo pacchetto viene sempre fortemente consigliato.

Però bisogna ricordare che questi filetti possono essere fonte di piccoli problemi se si decide di mettere uno sfondo colorato alla tabella, a singole celle, o a un'intera colonna o un'intera riga. Per evitare questi piccoli problemi bisogna ricorrere ai pilastrini esposti sopra, anche se questo richiede una maggiore attenzione da parte dell'utente/tastierista per non dimenticare nessun elemento compositivo da introdurre a mano.

8.5 Le tabelle di larghezza specificata

La tabella generata con l'ambiente asteriscato prevede che sia di larghezza specificata proprio mediante l'argomento ⟨*larghezza*⟩.

Per ottenere quella larghezza è necessario che le singole celle possano allargarsi quanto basta perché la tabella complessiva raggiunga il valore desiderato.

A questo scopo LaTeX dispone di una specificazione di una lunghezza 'elastica', detta anche 'gomma', che internamente è caratterizzata da tre valori, la lunghezza naturale, l'allungamento e l'accorciamento; se l'allungamento e l'accorciamento corrispondono a valori nulli, essa non si distingue da una lunghezza rigida, che è un altro tipo di lunghezza usato internamente da LaTeX.

Se invece l'allungamento e o l'accorciamento corrispondono a valori 'infiniti', si è in presenza di una 'gomma infinitamente allungabile o accorciabile'.

Nel caso delle tabelle si predispone allora una *@-espressione* che al suo interno, oltre agli altri eventuali elementi contenga anche la seguente espressione

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

Questa espressione inserisce una lunghezza infinitamente allungabile *in tutti i successivi* separatori di colonna, e questa gomma appare a destra degli altri separatori già presenti; ad esempio:

```
\begin{tabular*}{\textwidth}%
        {@{\extracolsep{\fill}\vline\hspace{\tabcolsep}}l|*3{c|}}
```

permette di definire una tabella con quattro colonne, delle quali la prima abbia tutte le celle con il loro contenuto allineato a sinistra e le altre tre abbiano le celle con il loro contenuto centrato; le quattro celle di ogni riga, quindi le quattro colonne, sono contornate da filetti verticali; la prima cella, inclusi i separatori, ha

le sue dimensioni naturali, mentre le celle successive hanno lo spazio a sinistra di ogni contenuto allargato di quanto basta per far sì che l'intera tabella sia larga quanto specificato.

Purtroppo questo modo di procedere va abbastanza bene per tabelle che quasi raggiungono naturalmente la larghezza desiderata, cosicché lo spazio che è necessario aggiungere sia otticamente trascurabile rispetto alla spaziatura normale. Invece se le celle hanno un contenuto abbastanza corto e la tabella deve essere allargata molto, il risultato è tutt'altro che ottimale. Si vedano le tabelle 8.3 e 8.4 per rendersi conto di quanto affermato. Si noti che nelle due tabelle citate sono presenti i filetti verticali che la buona tipografia non userebbe: sono stati inseriti per mostrare graficamente dove comincia e dove finisce ogni cella. Ma la tabella 8.3 rimane ben composta anche se si tolgono i filetti verticali; invece la tabella 8.4 rimane brutta anche se si tolgono i filetti verticali.

Famiglia			
Rossi Mastrodomenico Papadopoulos	Mario Giovanni Battista Filolaos	Maria Elisabetta Marta Penelopi	Filippo Elena Beatrice Irini

Tabella 8.3: Tabella di larghezza pari alla giustezza del testo ottenuta allargando poco le singole celle

Cognome	Età	Peso	Altezza
Rossi	50	85	182
Mastrodomenico	65	72	175
Papadopoulos	47	75	160

Tabella 8.4: Tabella di larghezza pari alla giustezza del testo ottenuta allargando troppo le singole celle

Ci sono diverse soluzioni a questo problema; una consiste nel richiamare il pacchetto di estensione `tabularx`, alla cui documentazione si rinvia il lettore.

Un'altra soluzione consiste nel definire un nuovo ambiente che calcoli la larghezza naturale delle singole celle e distribuisca omogeneamente lo spazio mancante a sinistra e a destra di ogni separatore di colonna, in modo che lo spazio bianco necessario per ottenere il risultato desiderato sia distribuito più uniformemente. Nella tabella 8.4 si osserverebbe allora che le colonne 2, 3 e 4, avrebbero davvero il loro materiale centrato, ma anche la colonna 1 avrebbe margini un poco più ampi così da ridurre lo spazio bianco nelle colonne di destra; si osservi il risultato nella tabella 8.5.

In ogni caso ci si ricordi che la presenza dei filetti verticali peggiora solamente l'effetto visivo, se bisogna davvero forzare un poco la composizione; per cui

Cognome	Età	Peso	Altezza
Rossi	50	85	182
Mastrodomenico	65	72	175
Papadopoulos	47	75	160

Tabella 8.5: Tabella di larghezza pari alla giustezza del testo ottenuta allargando lo spazio fra le singole celle

è quanto mai opportuno cercare di evitare l'inserimento dei filetti, in primis quelli verticali, possibilmente anche quelli orizzontali, accettando solo quelli che delimitano la tabella e il filetto sotto la prima riga che contiene le intestazioni delle colonne. Questa è la raccomandazione stilistica contenuta in moltissimi manuali di tipografia; è giusto che LaTeX consenta di inserire tutti i filetti che si vogliono, ma sta al compositore scegliere se metterli, quali mettere, e dove metterli.

Confesso: ho barato un pochino dicendo che basta usare un ambiente che calcoli la larghezza naturale delle singole celle e distribuisca omogeneamente lo spazio mancante a sinistra e a destra di ogni separatore di colonna. È vero; altrove ho generato e usato un simile ambiente (pacchetto *widetable*), ma nell'esempio della tabella 8.5 ho provato alcune volte modificando ad occhio il valore dello spazio di separazione intercolonna; mi sono bastati quattro o cinque tentativi per arrivare a definire che detto spazio, `\tabcolsep`, deve valere 8,9 mm; in questo modo, senza ricorrere a pacchetti esterni, la tabella 8.5 è stata composta con i comandi

```
\begin{table}[ht]
\def\V{\rule[-1ex]{0pt}{3.5ex}}
\def\S{\rule{0pt}{3,5ex}}
\setlength{\tabcolsep}{8.9mm}%  <---- semi spazio intercolonna
\begin{tabular}{|l|*3{c|}}
\hline
Cognome\V  & Età   &Peso  & Altezza  \\
\hline
Rossi\S    & 50    & 85   & 182      \\
...
\end{tabular}
\end{table}
```

Si noti che non si è usato l'ambiente *tabular**, ma il semplice ambiente *tabular*; la larghezza è stata ottenuta modificando lo spazio intercolonna finché l'occhio ha confermato che la tabella era larga quanto la giustezza.

Vale la pena di citare il pacchetto *tabu* alla cui documentazione si rimanda il lettore; questo pacchetto consente di gestire i colori sia dei filetti, sia del testo, sia dello sfondo delle celle; consente di agire come e meglio di *tabularx* per

generare una tabella di larghezza specificata, ma consente anche di allargare una data tabella di un ammontare specificato; consente l'uso delle celle di tipo `X`, tipiche dell'ambiente *tabularx* che le definisce, anche nella versione "long", cioè sia nell'ambiente *tabu* sia nell'ambiente *longtabu*; se in *tabu* o in *longtabu* si usano celle con i descrittori X, `p`, `m`, `b` il comando `\\` serve per terminare la riga della tabella, senza bisogno di ricorrere a `\tabularnewline`; consente l'incolonnamento basato sul separatore decimale, l'uso delle colonne `S` o `s` definite dal pacchetto *siunitx*; consente di colorare le righe di una tabella secondo colori definiti con i comandi del pacchetto *xcolor*, ma con comandi suoi propri che regolano meglio il riempimento dello sfondo anche in presenza dei filetti definiti con il pacchetto *booktabs*; insomma consente di fare molte delle cose che si potrebbero ottenere caricando diversi pacchetti (che il pacchetto *tabu* stesso emula o che vengono caricati dall'utente) ma consente anche di fare cose nuove che con quei pacchetti non si possono ottenere. Merita ricordare che l'ambiente *tabu* può essere annidato dentro *tabu* stesso, *tabular*, *tabular** e *tabularx* e viceversa; con alcune limitazioni lo si può annidare anche dentro le tabelle lunghe, ma è meglio riferirsi alla documentazione per valutare con attenzione queste limitazioni. L'ambiente *tabu* si può usare anche in modo matematico e può sostituire l'ambiente *array*. Nel capitolo 14 si mostra un esempio, equazione (14.33), il cui risultato non può essere ottenuto con gli altri ambienti e pacchetti LaTeX.

8.6 Problemi compositivi delle tabelle

Le tabelle sono difficili da comporre, non tanto perché i comandi necessari siano complessi, quanto perché esse devono venire progettate con anticipo, non devono essere composte sperando che vada tutto bene.

Ma anche con una buona impostazione il contenuto delle celle richiede aggiustamenti che non possono essere previsti in automatico dal programma, ma devono essere valutati con attenzione dal compositore.

Ecco quindi che nei prossimi paragrafi si daranno indicazioni per rimediare ad alcuni problemi che talvolta si presentano; questi suggerimenti, però, servano anche come stimolo per trovare autonomamente le soluzioni più consone ad altri problemi specifici che potrebbero manifestarsi.

8.6.1 Tabelle troppo larghe

Può darsi che una tabella sia così larga da fuoriuscire dalla giustezza; ma in questo caso si può rimediare in diversi modi a seconda che l'eccesso di larghezza rispetto alla giustezza sia di pochi punti, di pochi millimetri, o invece sia abbastanza consistente.

Nei primi due casi si possono mettere due *@-espressioni* 'vuote' come delimitatori a sinistra della prima cella e a destra dell'ultima cella; così facendo si eliminano le sporgenze dei filetti (se ci sono) fuori della giustezza della tabella.

Se ciò non bastasse, si potrebbe ancora ridurre lo spazio fra le colonne; questo spazio è dato da `\tabcolsep` che di default vale 6 pt. Si noti: questo spazio è quello che viene messo a sinistra e, rispettivamente, a destra di ogni separatore verticale; quindi i contenuti delle celle sono separati da uno spazio doppio, vale a dire da 12 pt. Non si perde molto nella composizione della tabella se si imposta il valore di `\tabcolsep` ad un valore di 3 o 4 pt; a seconda del numero delle colonne si possono recuperare diversi millimetri di larghezza.

Un'altra soluzione può consistere nel ridurre il corpo dei font con cui è composta la tabella; passando a `\small` o a `\footnotesize` spesso si risolvono problemi importanti. Basta procedere così:

```
\begin{table}
\small % oppure \footnotesize
\begin{tabular}{...}
...
\end{tabular}
\caption{...}
\label{tab:...}
\end{table}
```

Se il problema non è molto grave, sarebbe meglio evitare di scendere al corpo corrispondente a `\footnotesize`.

Simile a questa soluzione, esiste la possibilità di ricorrere al pacchetto *graphicx* che fornisce il comando `\resizebox`; questa soluzione è più flessibile, perché consente di scegliere il fattore di scala praticamente con continuità.

Resta il terzo caso dove la larghezza della tabella, nonostante le cure descritte nei capoversi precedenti, continua ad eccedere la giustezza del testo; se si tratta di una decina di millimetri al massimo, secondo il giudizio estetico del compositore, la si può collocare centrata lasciandola sporgere metà a sinistra e metà a destra. L'operazione è molto semplice, perché basta racchiudere tutta la tabella dentro una scatola di larghezza specificata; il comando `\makebox` che esegue questa operazione agisce secondo la seguente sintassi:

`\makebox[`⟨*larghezza*⟩`][`⟨*collocazione*⟩`]{`⟨*testo*⟩`}`

dove ⟨*larghezza*⟩ rappresenta la larghezza vera o apparente della scatola così creata; la ⟨*collocazione*⟩ specifica se il `{`⟨*testo*⟩`}` va collocato al centro, oppure accostato a sinistra o a destra dentro la scatola. Il ⟨*testo*⟩ può essere effettivamente del testo comune, ma può anche essere una tabella o qualunque altro oggetto già composto e trattato come un solo oggetto; anche una figura o un diagramma può essere usato come ⟨*testo*⟩. La ⟨*larghezza*⟩ è una larghezza vera se il ⟨*testo*⟩ ha una larghezza nominale inferiore a quella specificata, mentre essa è una larghezza apparente nel caso opposto; in altre parole se il ⟨*testo*⟩ è più largo di quanto specificato, lo si fa apparire largo esattamente come ⟨*larghezza*⟩.

Per centrare una tabella un pochino più larga della giustezza corrente basta dunque inserirla dentro una scatola larga quanto `\linewidth` specificandone la

posizione al centro della scatola (cioè non specificando nessuna ⟨*collocazione*⟩ visto che la posizione centrata è quella di default). Un costrutto del tipo

```
\begin{table}
\makebox[\linewidth]{%       Inizio scatola
   \begin{tabular}{...}
   ...
   \end{tabular}%
}%                           Fine scatola
\caption{...}\label{tab:...}
\end{table}
```

risolve il problema.

Tuttavia può ancora succedere che la tabella non possa essere 'aggiustata' con nessuno dei trucchi sopra esposti. Allora può succedere che l'eccesso di larghezza rispetto alla giustezza sia ancora dominabile con una rotazione di 90° in senso *antiorario*. Attenzione: la rotazione deve sempre avvenire in senso antiorario, indipendentemente dal fatto che la tabella venga poi fatta flottare in un recto invece che in un verso. Questo implica che per leggere la tabella, il documento debba venire ruotato sempre in verso *orario* di 90°.

Per eseguire questa rotazione, tenendo conto che la rotazione coinvolge anche la didascalia, è opportuno usare il pacchetto di estensione *rotating*, e al suo ambiente *sidewaystable*, alla cui documentazione si rinvia per reperire le necessarie informazioni d'uso.

8.7 Tabelle troppo lunghe

Con 'tabella lunga' si intende una tabella che non stia nell'altezza di una griglia di stampa. In questo caso ci sono diverse soluzioni, la prima delle quali è domandarsi se la tabella sia stata concepita bene; potrebbe darsi che scambiando le righe con le colonne la tabella diventi più larga e meno lunga; vale quindi la pena di ripensare alla possibilità di sistemare la tabella con questo semplice artificio.

Se la tabella dovesse risultare troppo larga, ma tale da poter stare in una sola pagina, si possono adottare i rimedi esposti nel paragrafo precedente.

Ma se la tabella è veramente lunga bisogna ricorrere a pacchetti di estensione, i più noti dei quali sono *longtable* e *supertabular* alla cui documentazione si rinvia, visto che questo tipo di composizione richiede un approccio abbastanza delicato.

Concettualmente il problema è semplice: si tratta di spezzare una lunga tabella in monconi tali da poter essere sistemati ciascuno in una pagina sola; per ovvie ragioni estetiche le colonne devono mantenersi della stessa larghezza in tutti i monconi, ma bisogna anche ripetere le intestazioni delle colonne, per non dover sfogliare pagine indietro per sapere se nella terza colonna compare, per esempio, la densità relativa oppure la massa volumica delle sostanze elencate nella tabella.

È opportuno anche che in ogni moncone della tabella che prosegue nella pagina successiva sia inserita una specie di intestazione di piè di pagina che specifica che la tabella continua; analogamente in ogni moncone della tabella, eccetto il primo, conviene mettere una intestazione dalla quale si capisca che si è nel cuore della tabella, non all'inizio né alla fine; questo può ottenersi anche ripetendo la didascalia alla fine della quale sia scritto 'continua', possibilmente fra parentesi e in corsivo, così da separare bene questa informazione dal resto della didascalia. Ovviamente il contatore della tabella non deve aumentare di una unità ad ogni moncone.

Quando si sono ben afferrati questi concetti, abbastanza semplici, allora diventa abbastanza facile comprendere la documentazione dei due pacchetti sopra indicati per potersi regolare al meglio e comporre una tabella professionale.

Un suggerimento non guasta. Conviene inserire il codice della lunga tabella in un file TeX a parte; solo il codice, senza `\documentclass` e senza l'ambiente *document*. Conviene fare uso del pacchetto *afterpage* e servirsi del comando `\afterpage`. Infatti, benché una lunga tabella non sia mobile, è opportuno che il primo moncone della tabella cominci all'inizio di una nuova pagina; mediante `\afterpage`, pertanto, diventa quasi spontaneo scegliere una posizione adeguata per dare il comando di leggere e di assorbire, usando il comando `\input`, il file che contiene il codice per la compilazione della tabella; questa automaticamente comincia all'inizio della prima pagina disponibile, finisce qualche pagina dopo, e il testo ricomincia subito dopo. Come si capisce bene, è estremamente importante scegliere il punto giusto per l'inserimento della lunga tabella; non necessariamente questo punto corrisponde alla fine di un capitolo; dipende dal genere di testo che si sta componendo. Se il codice per la lunga tabella è contenuto nel file `tabellalunga.tex`, memorizzato nella cartella `./file-inclusi`, allora quanto detto sopra corrisponde semplicemente a dare il comando

```
\afterpage{\input{file-inclusi/tabellalunga}}
```

Nel capitolo 24 si è fatto appunto così.

Si noti ancora che con le tabelle che si sviluppano su più pagine è necessario mettere la didascalia prima della tabella; per questo, a differenza dell'uso americano, rispecchiato in questo libro composto con la classe *book* alla quale non si sono apportate modifiche, sarebbe opportuno mettere sempre la didascalia prima di qualsiasi tabella, non solo prima di quelle lunghe. Per questo scopo, al fine di non dover aggiustare le cose a mano per ogni tabella che si compone, è possibile avvalersi del pacchetto *topcapt* che mette a disposizione del compositore il comando `\topcaption`, da usare esattamente come `\caption`; esso usa le spaziature giuste per mettere la didascalia sopra, invece che sotto, un qualunque oggetto flottante; si userà quindi `\topcaption` per le didascalie delle tabelle e `\caption` per le didascalie delle figure.

Nel prossimo paragrafo sarà mostrato un esempio d'uso delle potenzialità dei pacchetti di estensione per le tabelle, descrivendo quanto si è fatto per comporre le lunghe tabelle del capitolo 24.

8.8 Pacchetti di estensione per le tabelle

Oltre ai già citati pacchetti *tabularx*, *tabu*, *longtable* e *supertabular*, conviene riprendere il discorso del pacchetto *array* appena menzionato in relazione all'incolonnamento basato sul separatore decimale. Per dirla tutta, questo lavoro viene eseguito da una piccola estensione del pacchetto *array* che ne sfrutta appieno le potenzialità; si tratta del pacchetto *dcolumn* che è in dotazione standard di ogni distribuzione del sistema TEX.

Questo pacchetto definisce una colonna di nuovo tipo e con il codice `D`; ma accetta diverse personalizzazioni che è meglio esaminare nella documentazione.

Esercizio 8.3 Che cosa succederebbe alla tabella 8.2 se si usasse il descrittore `m{80mm}` oppure il descrittore `b{80mm}` al posto del descrittore `p{80mm}`?

Questi tre descrittori sono utilissimi, in particolare lo scrivente ha notato che, secondo i suoi gusti e il tipo di tabelle che compone, le celle con il blocco di testo centrato verticalmente sono più gradevoli rispetto a quando sono allineate sulla base della prima o dell'ultima riga. Ripeto, si tratta di una questione di gusti e, principalmente, del tipo di tabelle che si desidera comporre; in ogni caso è opportuno poter disporre di una scelta.

Qui si vuole richiamare l'attenzione del lettore sul fatto che il pacchetto *array* mette a disposizione del compositore due nuovi descrittori di colonna simili al descrittore `p{...}`; si tratta di `m{...}` e `b{...}`. Agiscono nello stesso modo, nel senso che tutti e tre i descrittori richiedono una larghezza di colonna da specificare al posto dei puntini; la differenza è che il contenuto della scatola di testo prodotto con `p{...}` ha la sua prima riga allineata con la prima o unica riga delle celle adiacenti; `b{...}` ha invece la sua ultima riga allineata con il contenuto delle celle adiacenti; infine `m{...}` è centrato con la sua riga mediana allineata con la riga mediana delle celle adiacenti. Vale la pena anche citare il fatto che uno degli aggiornamenti del pacchetto *array* (vedi il capitolo 30) ha recentemente avuto l'aggiunta dei descrittori di colonna `w` e `W` che, a differenza dei descrittori `p` e soci, richiedono due argomenti, con la sintassi:

`W{`⟨*posizione*⟩`}{`⟨*larghezza*⟩`}`

che vale anche per il descrittore `w`; la ⟨*larghezza*⟩ specifica la larghezza della cella, mentre la posizione specifica come il contenuto della cella è allineato dentro la ⟨*larghezza*⟩; valgono sempre i codici di posizione `l`, `c` e `r`; in sostanza sono colonne come `l`, `c` e `r` di ⟨*larghezza*⟩ specificata, non di larghezza che viene determinata dal programma di composizione in base al contenuto più largo dell'intera colonna. Si noti che `w` e `W` funzionano nello stesso modo, salvo che `w` non produce nessun messaggio se il suo contenuto straborda nella o nelle celle adiacenti perché è più largo della ⟨*larghezza*⟩ specificata, mente `W` produce un messaggio di avvertimento. Come sempre si trovano maggiori informazioni nella documentazione del pacchetto *array*.

È molto utile che il pacchetto *array* definisca il comando `\arraybackslash`; se l'ultima colonna ha il descrittore che esplicitamente o implicitamente contiene una delle dichiarazioni `\centering`, `\raggedright` oppure `\raggedleft`, l'ambiente di tabulazione non sa se un comando `\\` serva per andare a capo dentro la cella o se serva per terminare la composizione di una riga di celle; questo è particolarmente importante se si fa uso delle dichiarazioni di colonna introdotte con la sintassi descritta qui di seguito. Ecco allora che la dichiarazione `\arraybackslash` introdotta, per esempio, dopo `\centering`, permette di stabilire che il prossimo comando `\\` indica la terminazione della composizione della riga di celle. Con l'ambiente *tabu* questi accorgimenti sono superflui.

Il pacchetto *array* consente anche di comporre tabelle miste, in cui alcune colonne sono testuali ed altre puramente matematiche; anzi la caratteristica di *array* è quella di inserire una dichiarazione all'inizio di tutte le celle della stessa colonna e di porre una corrispondente dichiarazione alla fine di ciascuna di queste celle.

I descrittori delle colonne possono diventare particolarmente complessi, ma se si fa attenzione a descrivere con ordine ciascuna colonna, la cosa non è affatto complicata.

I nuovi 'comandi' per inserire queste dichiarazioni iniziali e finali in tutte le celle di una stessa colonna sono `>` e `<`. Essi seguono la sintassi seguente:

`>{`⟨*dichiarazioni iniziali*⟩`}`⟨*descrittore di colonna*⟩`<{`⟨*dichiarazioni finali*⟩`}`

Val più un esempio per descriverne l'azione, che non tante parole[1]; si predisponga una piccola tabella con le colonne allineate a sinistra, al centro, e a destra, ma dove la prima colonna deve avere tutti gli elementi in grassetto, la seconda deve contenere una descrizione in corsivo, la terza deve contenere un numero che rappresenta un prezzo e quindi deve essere presente l'unità monetaria; la tabella 8.6 è stata composta con i seguenti comandi

```
\begin{table}\centering
\begin{tabular}{>{\bfseries}l>{\itshape}cr<{\enspace\texteuro}}
\multicolumn3c{\textsf{\bfseries Prezziario}}\\\hline
Pantaloni   & alla zuava     & 35,00\rule{0pt}{2.5ex}\\
Pantaloni   & corti          & 12,50 \\
Camicie     & a scacchi      & 22,75 \\
Camicie     & botton down    & 31,20 \\
Calze       & di lana        &  8,33 \\
Calze       & di cotone      &  5,00 \\
Calze       & collant        & 15,70
\end{tabular}
\caption{...}\label{tab:arraypack}
\end{table}
```

[1] In questo stesso paragrafo si vedrà più avanti come è stata composta la tabella 24.8 facendo uso di questi descrittori speciali.

Prezziario		
Pantaloni	*alla zuava*	35,00 €
Pantaloni	*corti*	12,50 €
Camicie	*a scacchi*	22,75 €
Camicie	*botton down*	31,20 €
Calze	*di lana*	8,33 €
Calze	*di cotone*	5,00 €
Calze	*collant*	15,70 €

Tabella 8.6: Tabella composta con le estensioni del pacchetto `array`

e si vede quanto sia conveniente l'uso dei descrittori iniziali e di quelli finali per la composizione di colonne omogenee.

Ma una delle cose più utili del pacchetto `array` è costituito dalla possibilità di definire nuovi descrittori di colonna che possono ulteriormente semplificare la composizione delle tabelle.

Infatti il pacchetto `array` mette a disposizione il comando `\newcolumntype` con la seguente sintassi:

```
\newcolumntype{⟨descrittore⟩}{⟨specifiche⟩}
```

Il ⟨*descrittore*⟩ è formato da una sola lettera minuscola o maiuscola che non sia già stata usata per i descrittori esistenti e le ⟨*specifiche*⟩ sono costituite da un descrittore completo delle sue specificazioni particolari descritte mediante i comandi `>{...}` e/o `<{...}`; estendendo il concetto, il ⟨*descrittore*⟩ può corrispondere a delle ⟨*specifiche*⟩ che fanno uso di più descrittori.

Per esempio, nella tabella 8.6 sono stati usati per le tre colonne i descrittori

```
>{\bfseries}l>{\itshape}cr<{\enspace\texteuro}
```

Facendo uso del comando appena descritto si sarebbero potuti definire tre nuovi descrittori:

```
\newcolumntype{L}{>{\bfseries{l}}}
\newcolumntype{C}{>{\itshape{c}}}
\newcolumntype{E}{r<{\enspace\texteuro}}
```

e poi si sarebbe potuto descrivere il comando di apertura della tabella con:

```
\begin{tabular}{LCE}
```

È chiaro che se questi tipi di colonne vengono usati spesso, è conveniente eseguire le definizioni dei nuovi tipi di colonna una volta per tutte nel preambolo e poi usarli regolarmente nelle varie tabelle che si intendono comporre.

Al limite, se diverse tabelle usano la stessa sequenza di descrittori di colonna, questa può essere inclusa in un unico descrittore; per esempio

```
\newcolumntype{T}{LCE}
...
\begin{tabular}{T}
```

Anche nelle lunghe tabelle del capitolo 24 si è fatto nello stesso modo.

Per comodità del lettore, si riporta qui l'insieme delle dichiarazioni usate per comporre la tabella 24.8, compreso l'inizio e la fine della tabella, ma senza il contenuto, per ovvi motivi di brevità.

```
\newcommand*{\hstrut}{\rule{0pt}{3ex}\hskip 0sp}
\newcolumntype{B}{>{\bfseries}c}
\newcolumntype{M}{>{$\displaystyle}c<{$}}
\newcolumntype{R}{>{\raggedright}p{.35\textwidth}}
\newcolumntype{S}{>{\hstrut}p{.45\textwidth}<{\vspace*{1ex}}}
\newlength{\stretchandshrink}
\setlength{\stretchandshrink}{0pt plus 1fil minus 1fil}
...
\begingroup
\setlength\LTleft{0pt}
\setlength\LTright{0pt}
\begin{longtable}{@{\extracolsep{\stretchandshrink}}MRS@{}}%
    \caption{Simboli matematici\label{tmat}}\\
    \noalign{\smallskip\hrule\medskip}
    \multicolumn1B{Simbolo}&
    \multicolumn1B{Significato}&
    \multicolumn1B{Note}\\
    \noalign{\smallskip\hrule\medskip}
\endfirsthead
    \multicolumn3l{\emph{Continua tabella \thetable}}\\
    \noalign{\smallskip\hrule\medskip}
    \multicolumn1B{Simbolo}&
    \multicolumn1B{Significato}&
    \multicolumn1B{Note}\\
    \noalign{\smallskip\hrule\medskip}
\endhead
    \noalign{\smallskip\hrule\smallskip}
    \multicolumn3r{\emph{continua}}
\endfoot
    \noalign{\smallskip\hrule}
\endlastfoot
... & ... & ... \\% qui comincia la tabella vera e propria
\end{longtable}
```

Come si vede `\caption` permette di comporre la didascalia della lunga tabella al punto giusto, prima della tabella. `\endfirsthead` termina la descrizione necessaria per comporre le intestazioni delle colonne nel primo moncone. `\endhead` termina la descrizione necessaria per comporre l'intestazione delle colonne nei monconi successivi al primo; nello stesso tempo esso contiene anche l'informazione che la tabella che compare nella pagina corrente è una continuazione di una tabella iniziata in pagine precedenti e il riferimento `\thetable` serve appunto per ripetere il numero della tabella senza incrementarne il contatore. `\endfoot` termina la descrizione necessaria per comporre la fine di ogni moncone tranne

l'ultimo. `\endlastfoot` termina la descrizione necessaria per comporre la fine dell'ultimo moncone. I comandi per queste composizioni vengono memorizzati ed eseguiti ogni volta che ce ne sia la necessità, perché l'ambiente *longtable* provvede da solo a spezzare la tabella dove è necessario.

Fra i comandi usati si notano alcuni comandi primitivi, come `\noalign` che serve per inserire qualcosa dentro ad una tabella, senza tenere conto dei descrittori delle colonne; `\hrule` indica un filetto orizzontale che attraversa tutta la giustezza dello specchio di stampa; `\medskip` indica di inserire uno spazio verticale medio (circa mezza riga); `\smallskip` indica di inserire uno spazio verticale piccolo (circa di un quarto di riga). `\emph` serve per evidenziare una parola o una breve locuzione, nel senso che, se essa va evidenziata all'interno di un brano composto in tondo, esso evidenzia il suo argomento componendolo in corsivo; al contrario esso evidenzia in tondo all'interno di un brano in corsivo. La prima colonna è composta in modo matematico e in stile `\displaystyle`; la seconda colonna è composta in bandiera allineata a sinistra con una giustezza di 0,35 volte la giustezza della pagina; la terza colonna è composta con una giustezza pari a 0,45 volte la giustezza della pagina, ma solo dopo aver inserito un pilastrino definito come uno strut alto 3 ex seguito da uno spazio nullo[2]; questo spazio nullo rappresenta un trucchetto per assicurare che quanto segue sia eventualmente divisibile in sillabe (vedi capitolo 25). Il comando `\thetable` serve per scrivere il contenuto del contatore `table`.

Va ancora commentata l'impostazione delle due lunghezze `\LTleft` a sinistra della tabella e `\LTright` alla sua destra; esse sono collocate all'esterno della tabella, e aggiustandone i valori è possibile o lasciare alla tabella la sua larghezza naturale (basta che l'una e/o l'altra rappresentino lunghezze elastiche di allungabilità "infinita") sia, impostandone valori rigidi, allargare o restringere un pochino la tabella purché almeno una colonna sia allargabile o accorciabile a piacere; è quello che si è fatto nelle due lunghe tabelle del capitolo 24, ed è per questo che il primo separatore delle colonne è indicato con una *@-espressione* che contiene come argomento del comando `\extracolsep` una larghezza elastica `\stretchandshrink` di valore nominale nullo ma di allungabilità e di accorciabilità infinite (di primo ordine); perché questo? Perché in questo modo la tabella può *allargarsi o restringersi* a piacere.

L'ambiente *longtable* scrive nel file ausiliario le larghezze delle colonne di ogni moncone e richiede di comporre la tabella almeno tre volte per essere sicuri che tutti i monconi abbiano le colonne con le stesse larghezze; siccome ogni documento viene composto sempre diverse volte, se non altro per controllare le versioni in bozza prima di comporre la versione finale, questa necessità di comporre la tabella diverse volte in realtà è un prezzo piccolissimo da pagare per ottenere il risultato voluto. Inoltre è bene sapere che *longtable* non inizia mai una pagina nuova se non alla fine di una riga della tabella, quindi non può succedere che il testo di una cella risulti diviso fra due pagine consecutive.

[1.125] Nel paragrafo 19.10.3 sono esposti alcuni altri argomenti particolari relativi alle tabelle e alle figure inserite in alcune celle. Vi si definiscono dei nuovi comandi, ed è per questo che l'argomento è rinviato a quel paragrafo.

[2] L'unità di misura `sp` indica la frazione $1/2^{16}$ di un punto tipografico; la notazione in base 2 è chiarissima per un calcolatore, mentre la notazione decimale 1/65536 non è altrettanto significativa.

Nonostante il LaTeX3 Team abbia già predisposto i diversi pacchetti di estensione, descritti sopra, che consentono di migliorare la composizione delle tabelle e delle matrici, è opportuno segnalare una piccola lista, che non esaurisce tutte le possibilità disponibili negli archivi internazionali, ma che vale la pena di tenere presente in ogni caso.

Innanzi tutto i pacchetti *amsmath* e soci, prodotti dalla American Mathematical Society, estendono complessivamente le varie maniere di incolonnare oggetti, con riferimento specifico alle equazioni o a espressioni matematiche di vario genere; per questo pacchetto si rinvia la descrizione al capitolo 14.

L'Institution of Electrical and Electronics Engineers ha prodotto le sue classi per comporre gli articoli da pubblicare nelle sue riviste; ma ha messo a disposizione dei suoi autori anche un pacchetto separato *IEEEtrantools* che contiene diversi comandi e ambienti speciali, fra i quali gli ambienti per costruire tabelle e matrici in modo molto avanzato e non vincolato in modo così stretto alla sintassi tradizionale di LaTeX; questa infatti serve per 'mascherare' la complicazione della creazione della *riga modello* necessaria al comando primitivo `\halign` che è quello che effettivamente esegue poi la composizione delle tabelle e delle matrici. La documentazione si legge con `texdoc IEEEtrantools`, ma è preferibile leggerne le specifiche e gli esempi d'uso con `texdoc IEEEtran`.

Un altro pacchetto molto interessante è quello scritto (insieme ad altri utili pacchetti) da Mark D. Wooding, *mdwtab* con il suo corollario *mathenv*, che permettono di fare più o meno le stesse cose che si possono avere usando assieme i pacchetti *amsmath* e *array*. Alcune cose possono portare a risultati migliori che non con quei pacchetti, conservando però una grande versatilità e omogeneità di comportamento, in modo che con un unico ambiente per le tabelle si possono comporre sia tabelle vere e proprie, sia matrici matematiche, sia incolonnamenti di colonne miste; per le matrici vere e proprie sono disponibili ambienti appositi, in un certo senso più versatili di quelli, ottimi, messi a disposizione dal pacchetto *amsmath*.

Allo stesso tempo si possono garantire spaziature migliori sopra e sotto i filetti in modo che diventa veramente facile produrre tabelle o matrici con filetti attorno a tutte o a molte caselle con le spaziature giuste e senza che i filetti si interrompano o che i 'pilastrini' siano inefficaci. È vero che i filetti non dovrebbero essere usati per niente, eccettuati quelli che delimitano superiormente e inferiormente una tabella e il filetto che stacca le celle di intestazione dalle altre; tuttavia talvolta è necessario produrre tali tabelle, magari per replicare tabelle altrui anche nello stile compositivo, e i pacchetti di Wooding sono molto versatili.

Va notato, comunque, che per tabelle, matrici e sistemi di equazioni Wooding ridefinisce i comandi ordinari di LaTeX così che in effetti non c'è da documentarsi troppo, se non per imparare le infinite varianti e possibilità offerte dai suoi pacchetti.

Del pacchetto *tabu* e del suo ambiente *tabu* si è già detto in precedenza.

Capitolo 9

LaTeX: figure

9.1 Le figure e le immagini

Con LaTeX è opportuno distinguere bene fra 'figura' e 'immagine' o 'disegno' o 'grafico'; la *figura* è un oggetto non verbale (fotografia, disegno, grafico, schema, diagramma, eccetera) dotato di un numero e di una didascalia che LaTeX deve sistemare in qualche posto dove ci sia abbastanza spazio, visto che generalmente l'oggetto è di dimensioni relativamente grandi.

Invece l'*oggetto grafico* che viene trattato da LaTeX come una figura può avere le forme più disparate e può avere anche le origini più diverse. Potrebbe essere prodotto con una macchina fotografica, con un software da disegno, con i comandi stessi di LaTeX e dei suoi file di estensione; potrebbe anche essere costituito da codice di programmazione che verrà tradotto in grafico dal driver che rende leggibile agli umani il documento.

In questo capitolo si parlerà solo dell'ambiente *figure* e dei modi di produrre materiale grafico con LaTeX stesso, accennando appena ai disegni eseguibili con i file di estensione.

9.2 L'ambiente `figure`

L'ambiente *figure* è del tutto analogo all'ambiente *table*; serve per rendere flottante un oggetto e per dargli una didascalia il cui primo termine è il nome 'Figura', o 'Figure', o 'Abbildung' o ..., a seconda della lingua in uso, seguito dal suo numero progressivo e dalla didascalia vera e propria, composta come sempre da un titolo e da una spiegazione (facoltativa).

La sintassi è simile a quella dell'ambiente *table*:

```
\begin{figure}[⟨posizione⟩]
...
\end{figure}
```

e i parametri di ⟨*posizione*⟩ `h`, `t`, `b`, `p` e `!` sono gli stessi dell'ambiente *table*.

Fin qui la differenza rispetto all'ambiente *table* consiste solo nel fatto che viene scritto nel titolino la parola 'Figura' invece della parola 'Tabella'. In realtà il contatore delle figure è diverso da quello delle tabelle, così come la coda delle figure, o meglio, la gestione della coda delle figure è diversa e distinta da quella delle tabelle.

Le figure possono venire separate dal testo che precedono o che seguono da appositi filetti `\topfigrule`, `\botfigrule` o `\dblfigrule` descritti meglio nel capitolo 29. Normalmente questi filetti sono assenti, ma possono essere resi visibili e contribuiscono non poco alla separazione del testo dalla didascalia, specialmente per le figure collocate in testa alla pagina.

9.2.1 Controllo dei grandi oggetti flottanti

Ci si ricordi solo che oggetti troppo grandi, siano essi figure o tabelle, possono bloccare il deflusso degli oggetti della stessa specie dalla rispettiva coda e questo potrebbe portare non solo allo 'scarico' di tutti questi oggetti alla fine del capitolo o del documento, ma potrebbe portare anche alla saturazione delle code e quindi ad una situazione di stallo nella esecuzione della compilazione del documento.

Un modo di evitare questo stallo è quello di usare il pacchetto esterno *afterpage* al quale si è già fatto cenno nel capitolo precedente; esso mette a disposizione del compositore il comando `\afterpage` con la seguente sintassi:

```
\afterpage{⟨azione⟩}
```

dove ⟨*azione*⟩ rappresenta il comando da eseguire alla fine della pagina corrente, quando, cioè, la pagina corrente viene accodata al file di uscita. In quel momento, subito dopo l'accodamento, LaTeX può eseguire l'⟨*azione*⟩; nella fattispecie l'⟨*azione*⟩ prescritta potrebbe essere `\clearpage` che, oltre a terminare la pagina corrente (già terminata) serve anche per scaricare le code. Se si specifica spesso

```
\afterpage{\clearpage}
```

si ottiene il risultato di non accumulare troppo materiale nelle code delle figure e delle tabelle; magari durante la revisione delle bozze si può decidere se e come modificare figure o tabelle per rendere il deflusso dalle code un poco più ordinato in modo che non sia necessario ricorrere ad `\afterpage`.

9.2.2 Modifica degli ambienti flottanti

La rete e gli archivi CTAN contengono una moltitudine di pacchetti di estensione che consentono di gestire o di creare nuovi oggetti flottanti.

Il pacchetto principale per questo scopo è *float* la cui documentazione si trova in `.../doc/latex/float/float.pdf`. Esso tra le altre cose consente di usare un nuovo codice di posizionamento, `H`, che impone a LaTeX di collocare 'qui' l'oggetto 'senza fare tante storie'. Sembra che la possibilità di poter controllare

la posizione precisa degli oggetti flottanti sia un desiderio fortissimo degli utenti di LaTeX; in realtà, salvo pochi rari casi, è meglio lasciare fare a LaTeX il suo lavoro, anche se può sembrare un po' rigido nelle scelte che esso compie nel collocare gli oggetti flottanti; per capirne meglio il meccanismo conviene riferirsi al paragrafo 29.10 dove è descritto in dettaglio quello seguito dal programma di composizione per trovare la collocazione corretta degli oggetti flottanti. Agendo sui parametri che controllano queste posizioni si può fare molto meglio che non specificando il parametro `H`.[1]

Ma quel che più conta è che il pacchetto *float* consente di modificare le caratteristiche stilistiche degli ambienti flottanti e di definirne di nuovi; per esempio per esporre brani di programmazione, per descrivere degli algoritmi, insomma per collocare fuori testo qualunque cosa che possa essere reperita con un nome, un numero e una didascalia. Ognuno di questi ambienti flottanti può dare luogo a nuove liste/indici di oggetti flottanti alla stessa maniera con la quale si stampano gli elenchi delle figure e delle tabelle rispettivamente con i comandi `\listoffigures` e `\listoftables`.

Una ulteriore utile estensione è costituita dal pacchetto *floatrow*, la cui documentazione si trova in `.../doc/latex/floatrow/floatrow.pdf`; questo pacchetto consente di fare tutte le cose che si possono fare con il pacchetto *float* e molte altre ancora, al punto che se si usa *floatrow* non si deve caricare né *float* né *rotfloat* (quest'ultimo dedicato alla composizione di oggetti flottanti ruotati di 90°) perché il codice di questi pacchetti è stato già incorporato in *floatrow*. Quest'ultimo consente anche di collocare diversi (piccoli) oggetti flottanti, ognuno con la sua didascalia, in fila uno accanto all'altro (da cui il nome del pacchetto) e consente persino di inserire un (piccolo) oggetto flottante insieme al testo che lo affianca o che lo circonda.

Va detto che è giusto poter disporre delle possibilità descritte sopra, ma questi strumenti devono essere usati con molta cautela, perché il risultato, tecnicamente perfetto, potrebbe avere una estetica molto discutibile. Per esempio, se si vogliono inserire due figure, ognuna con la sua didascalia, in un unico ambiente flottante, è bene che esse abbiano le stesse dimensioni e che le didascalie abbiano lo stesso numero di righe; se così non fosse, l'aspetto complessivo delle due figure affiancate sarebbe esteticamente assai dubbio.

9.3 L'ambiente `picture`

LaTeX dispone di un meccanismo per produrre disegni al tratto, semplici schemi a blocchi, semplici diagrammi e simili disegnini mediante l'ambiente *picture*.

L'ambiente *picture* sta a *figure* come *tabular* sta a *table*; forse questo paragone serve a comprendere meglio la differenza fra figura e immagine.

L'ambiente *originale*, nato insieme al vecchio LaTeX 2.09, era molto limitato nella scelta delle linee che potevano essere tracciate, perché venivano usati dei segmentini tratti da una polizza di font specialmente predisposti per questo

[1] Detto in altri termini: *Non si usi mai il codice di posizione* `H`! Oppure: *Lo si usi pure solo se si sa quello che si sta facendo, ma non ci si lamenti se il risultato è pessimo*!

scopo; per quanto potessero essere ben disegnati, questi segmentini erano pur sempre in numero limitato.

Dal 2003 esiste l'estensione standard *pict2e* di LaTeX che non è ancora direttamente incorporata nel nucleo di LaTeX, ma che comunque fa parte degli strumenti obbligatori di qualunque installazione anche in una forma minima; questa estensione è progettata per eliminare tutte le limitazioni del vecchio ambiente e per aggiungere prestazioni nuove ed è per questo che nel seguito ci riferiremo solo a quanto si può fare con questo pacchetto. Si invocherà questa estensione mettendo nel preambolo il codice seguente al fine di disporre dell'attuale ultima revisione del codice (vedi il capitolo 30):

```
\usepackage[⟨opzioni⟩]{pict2e}[2020-09-30]
```

dove ⟨*opzioni*⟩ può essere una o più fra le opzioni disponibili, la maggior parte delle quali serve per definire il tipo di driver che verrà usato per tradurre il risultato in una forma comprensibile agli umani. Per lo più servono solo in casi particolarissimi e rari, perché in loro assenza il file capisce da solo di quale driver ha bisogno, ma ci sono altre tre opzioni interessanti:

original serve a impostare *pict2e* in modo che sia perfettamente equivalente al vecchio LaTeX; evidentemente serve solo per compatibilità con il passato.

ltxarrows serve per specificare a *pict2e* che le punte delle frecce devono essere disegnate come vengono disegnate da LaTeX in modo di compatibilità con il passato; siccome quelle frecce con punta triangolare leggermente concave sono molto gradevoli, il compositore può decidere di usare tali punte di freccia, anche se non introducono nessun elemento di novità nello stile del disegno.

pstarrows serve invece a disegnare le punte delle frecce come un aereo con le ali a V; questa è la maniera di disegnare le frecce usata anche dal pacchetto PSTricks, eccellente pacchetto di estensione che si affida completamente al driver dvips per produrre un file di tipo PostScript.

Se non si specificano opzioni, il pacchetto cerca se esiste il file di configurazione `pict2e.cfg` e sceglie le opzioni che quel file riesce a definire in base al programma eseguito; inoltre sceglie di default le frecce tradizionali di LaTeX.

A parte questi preliminari, l'ambiente *picture* definisce uno spazio grafico in cui l'unità di misura è definita dal parametro `\unitlength`, che di default vale 1 pt. Specificando un valore diverso è facilissimo scalare il disegno in modo da ingrandirlo o da rimpicciolirlo. Come trucco personale io preferisco definire per ogni ambiente *figure*, prima di aprire l'ambiente *picture*, una unità di misura dipendente dal font in uso, per esempio 1 ex, così che cambiando il corpo del font anche il disegno viene scalato proporzionalmente. In altre circostanze scelgo come unità di misura un centesimo della giustezza; in questo modo sono sicuro che, se non supero il valore di 100 unità in orizzontale, il disegno non esce fuori dei margini. La sintassi da usare per impostare l'unità di misura è:

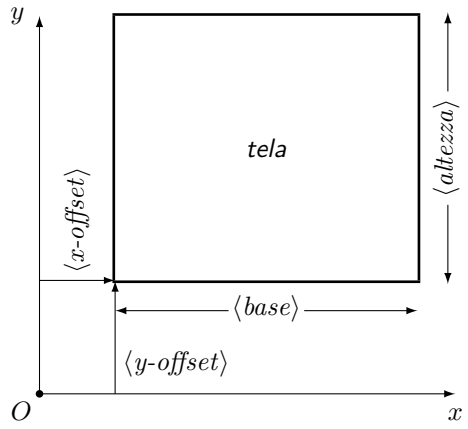


Figura 9.1: Relazione fra le dimensioni della tela e l'origine degli assi; i due offset possono anche essere negativi, così che l'origine degli assi cada dentro il rettangolo della tela.

```
\setlength{\unitlength}{⟨lunghezza⟩}
```

Dopo questa prima operazione l'ambiente *picture* con le sue dimensioni apparenti e il suo offset, nonché ogni distanza o lunghezza a cui si faccia riferimento all'interno di questo ambiente, è indicata mediante un numero, senza specificare le unità di misura, perché queste, è sottinteso, sono identificate da `\unitlength`.

La sintassi di apertura dell'ambiente *picture* è la seguente:

```
\begin{picture}(⟨base⟩,⟨altezza⟩)(⟨x-offset⟩,⟨y-offset⟩)
...
\end{picture}
```

Si notino le parentesi tonde per specificare i multipli dell'unità di misura che si intendono usare; la prima coppia di parentesi tonde contiene le dimensioni della base e dell'altezza 'apparenti' della 'tela' sulla quale si esegue il disegno; *apparenti* significa che il disegno verrà inserito dentro una tela rettangolare di quelle dimensioni, ma alcune parti del disegno potrebbero fuoriuscire da quel rettangolo, al punto che sarebbe possibile dichiarare entrambe le dimensioni pari a zero (e di fatto lo si fa spesso). La seconda coppia di parentesi è facoltativa; se c'è, essa indica la posizione dello spigolo inferiore sinistro della tela rispetto all'origine degli assi x e y, come si vede nella figura 9.1; questi sono gli assi rispetto ai quali vengono specificate le coordinate degli oggetti da collocare nel disegno.

Ogni oggetto da collocare nel disegno viene collocato mediante il comando `\put`; se l'oggetto deve essere collocato un certo numero di volte in posizioni regolarmente spaziate si può usare il comando `\multiput`; le sintassi sono le seguenti:

```
\put(⟨x⟩,⟨y⟩){⟨oggetto⟩}
\multiput(⟨x-ini⟩,⟨y-ini⟩)(⟨x-step⟩,⟨y-step⟩){⟨numero⟩}{⟨oggetto⟩}[⟨gestione
step⟩]
```

Le coordinate ⟨*x-ini*⟩ e ⟨*y-ini*⟩ sono le coordinate del primo punto; le coordinate ⟨*x-step*⟩ e ⟨*y-step*⟩ sono gli incrementi da dare ad x e ad y ogni volta che si deve inserire una nuova istanza dell'oggetto; il ⟨*numero*⟩ indica il numero totale di istanze dell'oggetto; infine ⟨*oggetto*⟩ è l'oggetto da collocare nel disegno. Per ⟨*gestione step*⟩ si vedrà più avanti; è una novità del pacchetto `curve2e` sviluppata nell'ottobre 2019. Ma senza usare questa nuova funzionalità (facoltativa), il comando `\multiput` si comporta come nelle precedenti versioni.

Gli oggetti collocabili sono linee, vettori (frecce), cerchi, dischi, testi semplici o riquadrati con linee continue o tratteggiate; i testi possono essere collocati intelligentemente rispetto al rettangolo ideale che li contiene. Possono poi essere collocati degli 'ovali', cioè dei rettangoli con gli spigoli arrotondati, dentro ai quali può essere collocato del testo; possono essere fatti semplici disegni tracciando delle curve mediante l'uso di curve di Bézier di secondo o di terzo grado. La sintassi che descrive ognuno di questi oggetti è la seguente:

```
\line(⟨x-pend⟩,⟨y-pend⟩){⟨lunghezza⟩}
\vector(⟨x-pend⟩,⟨y-pend⟩){⟨lunghezza⟩}
\circle{⟨diametro⟩}
\circle*{⟨diametro⟩}
\makebox(⟨base⟩,⟨altezza⟩)[⟨posizione⟩]{⟨testo⟩}
\framebox(⟨base⟩,⟨altezza⟩)[⟨posizione⟩]{⟨testo⟩}
\dashbox{⟨lung-trattino⟩}(⟨base⟩,⟨altezza⟩)[⟨posizione⟩]{⟨testo⟩}
\oval[⟨raggio⟩](⟨base⟩,⟨altezza⟩)[⟨parte⟩]
```

I nomi dei parametri dovrebbero essere abbastanza chiari, ma vanno specificate alcune cose particolari.

La lunghezza dei segmenti e dei vettori è la componente orizzontale, cioè la proiezione sull'asse x del segmento o del vettore; se questo è completamente verticale allora e solo allora la lunghezza coincide con quella del segmento o del vettore.

I coefficienti di pendenza dei segmenti e dei vettori rappresentano rispettivamente le proiezioni lungo gli assi x e y di un tratto di segmento o di vettore; in pratica sono i coseni direttori del segmento o del vettore, o quantità ad esse proporzionali (con lo stesso fattore di proporzionalità; per un segmento o un vettore che abbia una inclinazione di 60° rispetto all'orizzontale si possono specificare rispettivamente i valori 0.5 e 0.866; si otterrebbe lo stesso risultato se si specificasse 500 e 866; l'importante è che nessun coefficiente di pendenza superi in valore assoluto il valore 16383.99999, a causa del modo particolare che usa il programma di compilazione per eseguire i suoi calcoli interni. Per i segmenti, che non hanno una punta di freccia che ne indichi la direzione, questa va intesa come la direzione del segmento dal suo estremo collocato con `\put` all'altro estremo;

in altre parole il comando `\put` colloca alla sua coordinata il primo punto del segmento, da cui parte il segmento con la pendenza specificata.

Il comando `\circle` disegna una circonferenza completa di cui si specifica il ⟨*diametro*⟩; il suo punto di riferimento messo in posizione con `\put`, è il suo centro. Il comando `\circle*` specifica invece un disco completo, non solo il contorno, quindi disegna un cerchio 'nero' con il ⟨*diametro*⟩ specificato.

I comandi `\makebox`, `\framebox` e `\dashbox` definiscono tre scatole, la prima senza che ne venga disegnato il contorno, la seconda con il contorno disegnato, la terza con il contorno tratteggiato con trattini lunghi ⟨*lung-trattino*⟩. Il comando `\put` mette in posizione il loro punto di riferimento, cioè lo spigolo inferiore sinistro. All'interno di questi rettangoli, o scatole, può essere posto del testo che viene collocato rispetto ai bordi ideali, disegnati o tratteggiati conformemente ai parametri di ⟨*posizione*⟩; questi sono le solite lettere `t`, `b`, `l`, `r` e `c`, con il rispettivo significato di 'top', 'bottom', 'left', 'right' e 'center'; per ogni oggetto si possono specificare al massimo due parametri di posizione[2], ricordando che 'center' è sempre il valore di default.

È particolarmente comodo il comando `\makebox` con dimensioni di ⟨*base*⟩ e ⟨*altezza*⟩ nulle, perché il ⟨*testo*⟩ che vi viene inserito, risulta collocato con precisione rispetto allo spigolo inferiore sinistro, ma senza spazio alcuno interposto; non ha importanza che le dimensioni apparenti della scatola siano nulle; è importante che il testo sia collocato con un riferimento preciso, indipendentemente dalla presenza di ascendenti o discendenti.

Per il comando `\oval` è possibile specificare il raggio minimo del quarto di cerchio che costituisce lo spigolo; le dimensioni di ⟨*base*⟩ e ⟨*altezza*⟩ di questo speciale rettangolo a spigoli arrotondati si riferiscono al rettangolo completo; ma la specifica facoltativa di ⟨*parte*⟩ permette di scegliere quali angoli mostrare o quale coppia di angoli adiacenti mostrare, così che specificando raggio, metà della base e metà dell'altezza uguali è possibile disegnare solo un quarto di circonferenza oppure metà circonferenza; al solito la ⟨*parte*⟩ è specificata con le lettere `l`, `t`, `b` e `r` (`c` non avrebbe senso), cosicché `t` permette di disegnare solo la metà di sopra, `tl` solo il quarto in alto a sinistra.

Nell'ambito dell'ambiente *picture* è possibile specificare due spessori predefiniti per le linee da tracciare, `\thinlines` e `\thicklines`; si può specificare un valore assoluto di spessore (espresso con le unità di misura) mediante:

```
\linethickness{⟨spessore assoluto⟩}
```

e questo comando modifica lo spessore sia delle linee rette e comunque inclinate, sia delle linee curve, siano esse archi di circonferenza oppure curve di Bézier quadratiche o cubiche.

Infine i comandi `\qbezier` e `\cbezier` consentono di mettere in posizione delle curve di secondo o di terzo grado rispettivamente, dette *spline* di Bézier,

[2] Ovviamente non contrastanti. È ovvio che non si possono specificare simultaneamente `t` e `b`, perché lo stesso oggetto non può essere collocato allo stesso tempo in testa e al piede dentro la scatola specificata.

senza bisogno di ricorrere ai comandi `\put` o `\multiput`. La loro sintassi è la seguente:

```
\qbezier(⟨x1⟩,⟨y1⟩)(⟨x2⟩,⟨y2⟩)(⟨x3⟩,⟨y3⟩)
\cbezier(⟨x1⟩,⟨y1⟩)(⟨x2⟩,⟨y2⟩)(⟨x3⟩,⟨y3⟩)(⟨x4⟩,⟨y4⟩)
```

Per la curva di secondo grado, descritta da `\qbezier`, (x_1, y_1) rappresentano le coordinate del punto da cui la curva parte in direzione del punto avente le coordinate (x_2, y_2); invece (x_3, y_3) rappresentano le coordinate del punto di arrivo con la direzione che proviene da (x_2, y_2).

Per la curva di terzo grado, descritta da `\cbezier`, essa parte da (x_1, y_1) in direzione (x_2, y_2) e arriva al punto (x_4, y_4) con la direzione proveniente da (x_3, y_3).

Scegliendo accuratamente i punti iniziali e finali, nonché le direzioni delle tangenti si possono facilmente disegnare semplici diagrammi come mostrato nelle figure 9.2 e 9.3. I punti guida delle curve di Bézier non compaiono perché cadono fuori dal rettangolo della figura.

Va notato che nel 2009 il pacchetto *pict2e* è stato fortemente arricchito di nuove funzionalità che non sono delle semplici aggiustatine agli stessi comandi della versione tradizionale di LaTeX, ma sono delle estensioni decisamente importanti.

Innanzi tutto è stata definita una nuova routine di divisione fra numeri fratti che permette di dividere numeri di qualsiasi grandezza (purché inferiori in modulo a 16 384, sia dividendo, sia divisore, sia quoziente); per cui la restrizione sul fatto che inizialmente (nella versione di *pict2e* del 2004) i coefficienti di pendenza di segmenti e vettori dovessero essere rappresentati da numeri interi inferiori in modulo a 1000, è stata rimossa; ora si possono usare tranquillamente numeri fratti e, se per esempio si vuole una pendenza di 37°, il cui coseno vale 0,79864 e il cui seno vale 0,60182, si possono usare direttamente questi due numeri senza dover ricorrere all'artificio di moltiplicarli per 1000 e poi usare la sola parte intera arrotondata; non cambia molto da un punto di vista pratico, ma evita di dover fare calcoli e arrotondamenti.

Inoltre sono stati aggiunti altri comandi, alcuni dei quali tracciano il loro disegno quando siano messi come argomento di `\put`, e altri che sono assoluti, e cioè, come per le curve di Bézier, non hanno bisogno dell'intermediario del comando `\put`.

Comandi da usare con `\put`:

`\arc[`⟨*angolo1*⟩`,`⟨*angolo2*⟩`]{`⟨*raggio*⟩`}`
`\arc*[`⟨*angolo1*⟩`,`⟨*angolo2*⟩`]{`⟨*raggio*⟩`}`

Comandi che non necessitano di `\put`:

`\Line(`⟨x_1⟩`,`⟨y_1⟩`)(`⟨x_2⟩`,`⟨y_2⟩`)`
`\polyline(`⟨x_1⟩`,`⟨y_1⟩`)...(`⟨x_n⟩`,`⟨y_n⟩`)`
`\polygon(`⟨x_1⟩`,`⟨y_1⟩`)...(`⟨x_n⟩`,`⟨y_n⟩`)`
`\polygon*(`⟨x_1⟩`,`⟨y_1⟩`)...(`⟨x_n⟩`,`⟨y_n⟩`)`

I comandi `\arc` con o senza asterisco generano un arco con il centro nella coordinata specificata dalle coordinate del comando `\put`, di raggio pari a ⟨*raggio*⟩, che inizia nell'angolo ⟨*angolo1*⟩ e termina nell'angolo ⟨*angolo2*⟩; gli angoli sono specificati in gradi e con il segno positivo si indica la rotazione in senso antiorario. La versione asteriscata riempie di colore (di default nero) il settore sotteso dall'angolo.

Il comando `\polyline` genera una spezzata che collega in successione gli n punti (x_1, y_1), (x_2, y_2), ..., (x_n, y_n); il numero di punti è arbitrario, perché il comando itera il proprio funzionamento fino ad esaurire la lista delle coordinate.

Il comando `\polygon` genera una spezzata chiusa nel senso che arrivato al punto n-esimo aggiunge ancora un segmento fino il primo punto della lista. Il comando `\polygon*` si comporta nello stesso modo ma riempie di colore (di default nero) l'interno del poligono.

Inoltre sono stati definiti nuovi comandi per specificare come devono essere formati gli estremi dei segmenti (tagliati dritti o arrotondati) e come debbano essere raccordati fra di loro i segmenti che formano le spezzate (se si prolungano fino ad incontrarsi i lati che delimitano i due segmenti da unire, oppure se questo prolungamento viene troncato ad una certa distanza dal punto di intersezione dei loro assi); per i dettagli si veda la documentazione. Si noti che con linee sottili questi dettagli sono virtualmente impercettibili, ma con linee relativamente spesse la differenza c'è e si vede. Ricordiamoci che con *pict2e* si possono variare a piacere gli spessori di tutte le linee, quindi non è difficile ottenere segmenti di un certo spessore; semplicemente si cerchi di non dimenticare che una linea di uno spessore di circa un millimetro ($\approx 1{,}5\,\text{pt}$) è una linea nerissima; si vedano le figure 9.4 e 9.5; in quest'ultima si notino le terminazioni delle linee e i raccordi per fare segmenti di una spezzata eseguite con i comandi indicati nella figura stessa; i segmenti sono molto spessi per rendere evidenti a occhio nudo gli effetti di quei comandi.

Usando i colori e i nuovi comandi di *pict2e* non è difficile disegnare il diagramma della figura 9.4.

Il codice per disegnare la figura 9.4 è il seguente:

```
\begin{figure}[!t]\unitlength=0.85cm\centering
\begin{picture}(11,11)(-1,-1)
\linethickness{0.35pt}
```

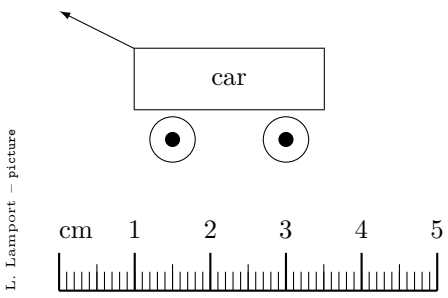


Figura 9.2: Il semplice disegno usato da Leslie Lamport per descrivere le potenzialità dell'ambiente *picture*

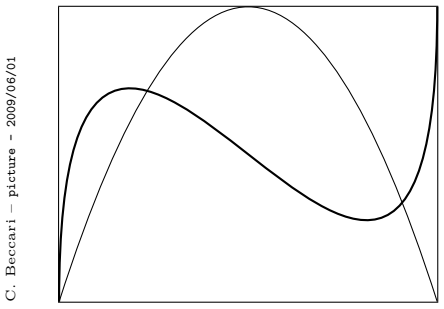


Figura 9.3: Alcune curve di Bézier di secondo e di terzo grado tracciate nell'ambiente *picture*

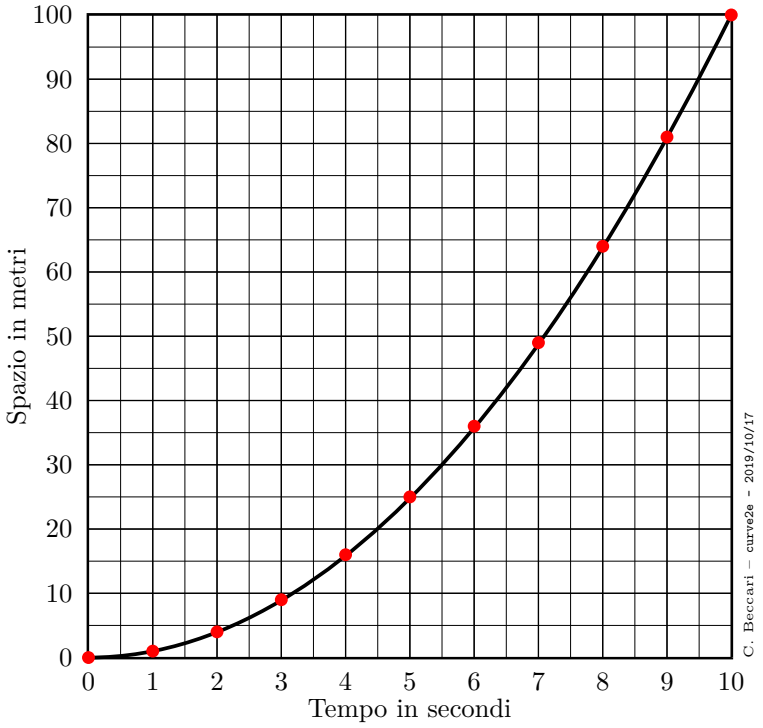


Figura 9.4: Moto uniformemente accelerato

```
\multiput(0,0)(0.5,0){21}{\line(0,1){10}}
\multiput(0,0)(0,0.5){21}{\line(1,0){10}}
\linethickness{0.5pt}
\multiput(0,0)(1,0){11}{\line(0,1){10}}
\multiput(0,0)(0,1){11}{\line(1,0){10}}
```

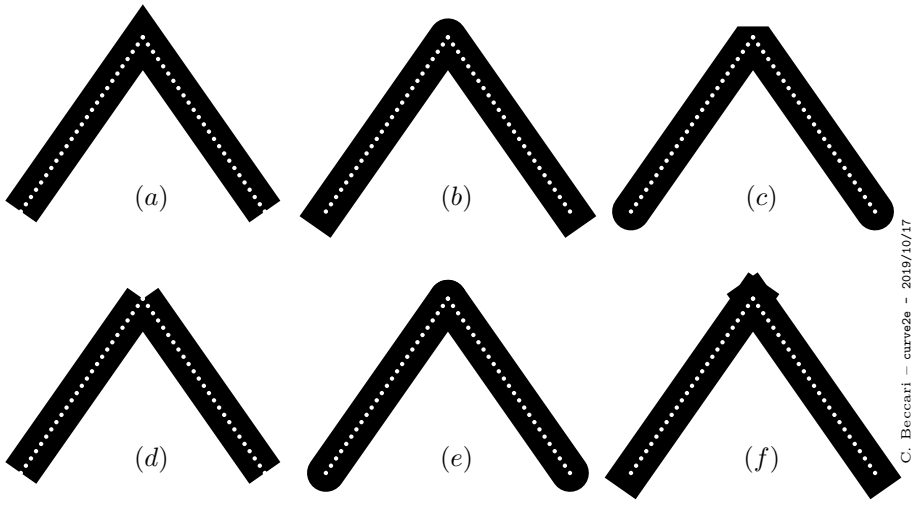


Figura 9.5: Le tre giunzioni e le tre terminazioni delle linee spesse. Giunzioni: (*a*) miter joint, `\miterjoin`; (*b*) round joint, `\roundjoin`; (*c*) bevel joint, `\beveljoin`. Terminazioni: (*a*) e (*d*) butt cap, `\buttcap`; (*b*) e (*e*) square cap, `\squarecap`; (*c*) e (*f*) round cap, `\roundcap`. Le linee di puntini marcano l'asse delle righe spesse e congiungono esattamente i punti da cui partono e arrivano i segmenti spessi. Le spezzate della linea superiore sono eseguite con `\polyline`, mentre nella riga inferiore sono usati singoli segmenti. Si vede chiaramente che con le linee spesse, due segmenti connessi con le terminazioni di tipo "butt cap" o "square cap" presentano un gradino o una sporgenza (se i due segmenti formano un angolo diverso da 90°), mentre se la terminazione è di tipo "round cap" la giunzione non presenta difetti.

```
\linethickness{0.7pt}
\setcounter{cms}{0}
\put(0,0){\framebox(10,10){}}
\multiput(0,-0.5)(1,0){11}{%
  \makebox(0,0)[b]{\arabic{cms}}\stepcounter{cms}}
\put(10.2,0){\rotatebox{90}{\tiny C. Beccari --
  \texttt{curve2e - 2019/10/17}}}
\put(5,-1){\makebox(0,0)[b]{Tempo in secondi}}
\setcounter{cms}{0}
\multiput(-0.25,0)(0,1){11}{%
  \makebox(0,0)[r]{\arabic{cms}%
  \addtocounter{cms}{10}}}
\put(-0.9,5){\rotatebox{90}{%
  \makebox(0,0)[b]{Spazio in metri}}}
\linethickness{1.4pt}
\cbezier(0,0)(4.5,0)(8,6)(10,10)
\color{red}
```

```
\multiput(0,0)(0,0){11}{\circle*{0.2}}%
  [\edef\X{\fpeval{\X+1}}%
  \edef\Y{\fpeval{(\X**2)/10}}]
\end{picture}
\caption{Moto uniformemente accelerato}
\label{fig:motoaccelerato}
\end{figure}
```

Vi sono molti aspetti un po' particolari, come per esempio l'uso delle scatole `\makebox` con le dimensioni dichiarate nulle; come già detto, questo è un espediente che si usa spesso e serve per poter collocare il contenuto della scatola in una posizione precisa rispetto ai suoi bordi (invisibili), mantenendo nello stesso tempo un completo controllo sul punto di riferimento dell'oggetto da mettere in posizione o anche da ruotare, come la legenda della scala delle ordinate.

Per sovrapporre i puntini rossi sopra la curva si è usato un aggiornamento del pacchetto *curve2e* che permette di fare una cosa impossibile con la definizione originale. A parte il fatto che le coordinate dei comandi `\put` e `\multiput` consentono sia la forma cartesiana (x, y), sia la forma polare $(\alpha; \rho)$, il comando `\multiput` ora accetta un argomento facoltativo ⟨*gestione step*⟩ che agisce in modo del tutto libero all'esterno ai gruppi o alle scatole, che mettono in posizione l'⟨*oggetto*⟩; le due componenti del comando di posizionamento che si trova dentro il ciclo di ripetizione controllato dal numero di ripetizioni fornito come terzo argomento del comando, si chiamano `\X` e `\Y`. Il controllo si esegue assegnando a queste due macro le espressioni che si desiderano mediante le "funzioni" del linguaggio L3, contenute nell'argomento di `\fpeval`.

La "funzione" `\fpeval`[3] serve per calcolare espressioni matematiche anche con numeri fratti o in notazione scientifica; qui sembra sprecato essendo usato semplicemente per calcolare un quadrato di numeri interi, ma il suo uso è necessario perché si tratta di un comando sviluppabile, usabile, quindi, dentro l'argomento facoltativo della macro `\multiput`. In questo caso, l'ascissa viene incrementata di una unità, mentre all'ordinata si assegna il quadrato dell'ascissa diviso per dieci per tenere conto delle scale diverse dei due assi.

Invece il codice per disegnare la figura 9.5 è il seguente. Vi si possono notare certi usi insoliti degli argomenti del comando `\put`; si noti l'uso del colore bianco per tracciare le linee punteggiata a marcare gli assi dei tratti scuri.

```
\begin{figure}[!htb]
\centering\unitlength=0.01\textwidth
\begin{picture}(100,55)(0,2)
```

[3] Questo comando richiede che si lavori con una installazione del sistema TeX almeno del 2018, e che sia stato caricato il pacchetto *xfp*. Nel file di macro usato per comporre questa guida (`MacroGuida.sty`) questo caricamento non appare esplicitamente, perché ci pensa il pacchetto *cureve2e*, che estende le funzionalità dell'ambiente *picture*; una di queste funzionalità, per esempio, è il tracciamento delle linee punteggiate visibili nella figura 9.5, eseguito con il comando `\Dotline`. Un'altra funzionalità è fornita proprio con questo esempio dove si tracciano alcuni punti di una parabola.

```
\put(101,6){\rotatebox{90}{\tiny C. Beccari --
   \texttt{curve2e - 2019/10/01}}}
\linethickness{5mm}
\put(0,30){%
  \put(0,5){\buttcap\polyline[\miterjoin](0,0)(14,20)(28,0)
  \put(15,0){\makebox(0,0)[b]{$(a)$}}}
  \put(35,5){\squarecap\polyline[\roundjoin](0,0)(14,20)(28,0)
  \put(15,0){\makebox(0,0)[b]{$(b)$}}}
  \put(70,5){\roundcap\polyline[\beveljoin](0,0)(14,20)(28,0)
  \put(15,0){\makebox(0,0)[b]{$(c)$}}}
  {\color{white}%
    \put(0,5){\Dotline(0,0)(14,20){1}\Dotline(14,20)(28,0){1}}
    \put(35,5){\Dotline(0,0)(14,20){1}\Dotline(14,20)(28,0){1}}
    \put(70,5){\Dotline(0,0)(14,20){1}\Dotline(14,20)(28,0){1}}
  }
}
%
\put(0,5){\buttcap\Line(0,0)(14,20)\Line(14,20)(28,0)
\put(15,0){\makebox(0,0)[b]{$(d)$}}}
\put(35,5){\roundcap\Line(0,0)(14,20)\Line(14,20)(28,0)
\put(15,0){\makebox(0,0)[b]{$(e)$}}}
\put(70,5){\squarecap\Line(0,0)(14,20)\Line(14,20)(28,0)
\put(15,0){\makebox(0,0)[b]{$(f)$}}}
{\color{white}%
  \put(0,5){\Dotline(0,0)(14,20){1}\Dotline(14,20)(28,0){1}}
  \put(35,5){\Dotline(0,0)(14,20){1}\Dotline(14,20)(28,0){1}}
  \put(70,5){\Dotline(0,0)(14,20){1}\Dotline(14,20)(28,0){1}}
}
\end{picture}
\caption[...]{...}
\label{fig:join}
\end{figure}
```

Insomma, l'ambiente *picture* è semplice, ma con le estensioni introdotte con il pacchetto *pict2e*, e ancor più con il pacchetto *curve2e*, non è inferiore a molti altri software di disegno; ha il vantaggio che ogni informazione letterale o matematica che compare nel disegno è eseguita con gli stessi caratteri usati nel testo. Naturalmente non è l'unico mezzo per ottenere questo risultato, come d'altra parte si ricava da quanto esposto nel prossimo paragrafo.

Il pacchetto *curve2e* è stato esteso ulteriormente con un altro pacchetto, *euclideangeometry*, con il quale si possono eseguire i disegni (e i calcoli) con riga, squadra e compasso, alla stessa maniera con cui viene insegnata la geometria euclidea nelle scuole secondarie inferiori. Per *curve2e* e *euclideangeometry* i manuali d'uso, ricchi di esempi, dal 2010 sono distinti dalle rispettive documen-

tazioni dei loro codici; l'utente è invitato, perciò, a leggere tali manuali d'uso con i comandi `texdoc curve2e-manual` e `texdoc euclideangeometry-man`.

9.4 Il pacchetto pgf

Il pacchetto di estensione *pgf* è ancora in evoluzione ma certamente le potenzialità che ha oggi verranno sicuramente mantenute ad ampliate in futuro. L'autore cerca di costruire una interfaccia utente che prescinda completamente dal particolare strumento informatico che trasforma il codice in qualcosa comprensibile dagli umani. Oggi si possono ottenere risultati ottimi con il pacchetto di estensione *PSTricks*, ma il suo codice è interpretabile solo dal driver dvips; per terminare con un file in formato PDF è necessario procedere ad una ulteriore conversione di formato[4].

L'acronimo PGF indica 'Portable Graphics Format' e vorrebbe poter un giorno essere altrettanto potente di *PSTricks* ma svincolato da questo o quel driver; nel momento in cui il sorgente `.tex` viene compilato, il programma di compilazione determina da chi questa compilazione viene eseguita e sceglie l'interfaccia giusta in relazione a quella situazione; se il file sorgente, come succede spesso, viene spedito per posta elettronica ad un conoscente che lavora con una piattaforma diversa e una distribuzione del sistema TeX diversa o è abituato ad usare un compilatore diverso, ebbene la grafica deve risultare del tutto identica a quella che otteneva l'autore originale.

Qui sarebbe fuori luogo scendere nei dettagli, e si rinvia il lettore alla documentazione del pacchetto *pgf*; sappia comunque il lettore che questo pacchetto mette a disposizione un ambiente di disegno che si chiama *tikzpicture*, all'interno del quale la sintassi dei comandi è molto diversa dalla sintassi da usare nell'ambiente *picture*, ma sotto certi aspetti essa è più intuitiva, in ogni caso più adatta alla moltitudine di oggetti, linee, colori, nodi, alberi, strutture, schemi di flusso, eccetera, che possono essere gestiti con quel pacchetto e quell'ambiente.

Qui, giusto per far venire l'acquolina in bocca con un blando aperitivo, si mostra solo la figura 9.6, dove i colori delle sfere che giacciono sulla diagonale secondaria (quella che va da SW a NE) sono mescolati fra di loro nelle posizioni fuori da quella diagonale, mostrando sempre delle mezze sfere con gradienti di luce obliqua a simmetria circolare.

L'ambiente *tikzpicture* è estensibile con diversi altri ulteriori pacchetti; per esempio, *circuitikz* che permette di disegnare circuiti elettronici scegliendo lo stile americano oppure quello europeo; la figura 9.7 mostra un esempio tratto dal manuale del pacchetto che si trova in `.../doc/latex/circuitikz`. Il manuale

[4]Questa limitazione, che provoca una apparente incompatibilità fra il compilatore latex e i compilatori pdflatex, xelatex e lualatex, può essere aggirata attraverso il pacchetto *pst-pdf*, alla cui documentazione si rinvia il lettore; un altro pacchetto *auto-pst-pdf* offre una comoda interfaccia per l'uso di *pst-pdf* così che di fatto, sia pure con un lieve rallentamento della compilazione, anche i tre compilatori riescono a gestire file sorgente che fanno uso di *PSTricks*. Il lettore interessato può esplorare CTAN`/macros/latex/contrib/pst-pdf/` e CTAN`/macros/latex/contrib/auto-pst-pdf/`.

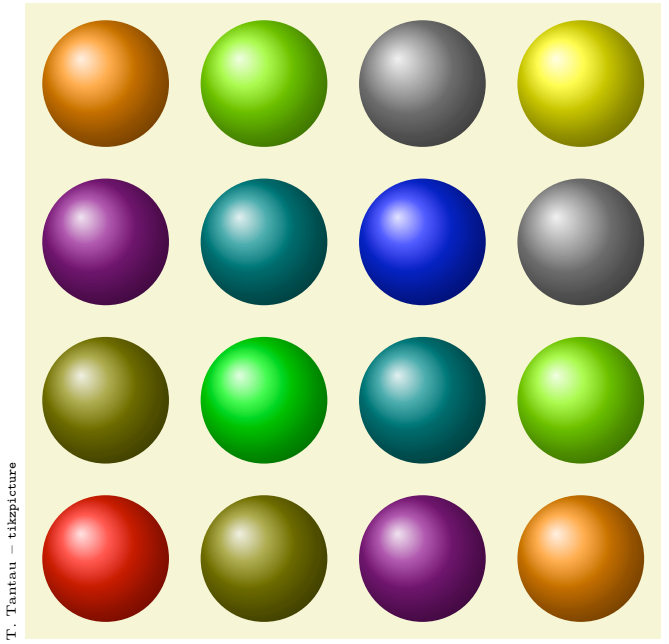


Figura 9.6: Un disegno a colori ottenuto con l'uso del pacchetto *pgf* e l'ambiente *tikzpicture*

espone anche l'intera lista di componenti, veramente molto ricca, con cui si possono disegnare circuiti elettronici; essendo ogni componente disegnato mediante una macro, non è difficile, casomai ce ne fosse bisogno, arricchire questa enorme collezione di ulteriori segni particolari.

L'ambiente *tikzpicture* e i comandi del pacchetto *tikz* possono venire usati per la definizione di diverse macro; in questo testo esso è stato usato per definire le macro che disegnano le icone che compaiono sui tasti di controllo della tastiera del portatile Mac Book Pro; esse possono venire esaminate nel file `MacroGuida.sty`.

Un altro semplice esempio suggerito da Antonio Cervone è il disegno per definire una paginetta con l'angolo superiore ripiegato "morbidamente"; questo disegno appare anche nell'Arte di scrivere con LaTeX, [55]. Se ne è rielaborato il codice proposto da Antonio Cervone in modo da creare una macro che funziona solo con una versione moderna di pdflatex perché vi vengono usate delle funzionalità estese. Il codice rielaborato è il seguente:

```
\newlength\paginettaunit

\newcommand\paginetta[2][.5\textwidth]{{%
\countdef\Y=1000\countdef\YY=1001\relax
\setbox0\vbox{\hsize=#1\relax#2}\relax
\dimen0=#1\relax\dimen0=1.2\dimen0\relax
```

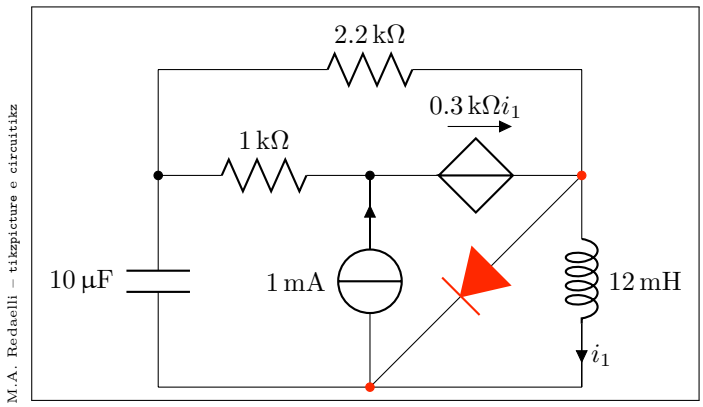


Figura 9.7: Circuito elettronico disegnato usando l'ambiente *tikzpicture* avendo caricato l'estensione *circuitikz*

```
\paginettaunit=\dimexpr\dimen0/100\relax
\dimen0=\ht0
\Y=\numexpr\dimen0/\paginettaunit\relax
\YY=-\numexpr\Y+40\relax
\Y=-\numexpr\Y/2+27\relax
\begin{tikzpicture}[thick,x=\paginettaunit,y=\paginettaunit]
\draw[fill=blue!10]
  (0,0) [rounded corners=7.5] --
  (80,0) --
  (100,-20) [rounded corners=0] --
  (100,\YY) --
  (0,\YY) --
  cycle;
\draw
  (75,0) .. controls (80,0) and (80,-5) ..
  (80,-20) .. controls (95,-20) and (100,-20) ..
  (100,-25);
\node at (50,\Y) {\box0};
\end{tikzpicture}}}
```

e il risultato dell'uso del comando:

```
\paginetta[.55\textwidth]{\raggedright
Questo è un testo composto da due capoversi.

Infatti questo è il secondo capoverso.}
```

appare nella figura 9.8.

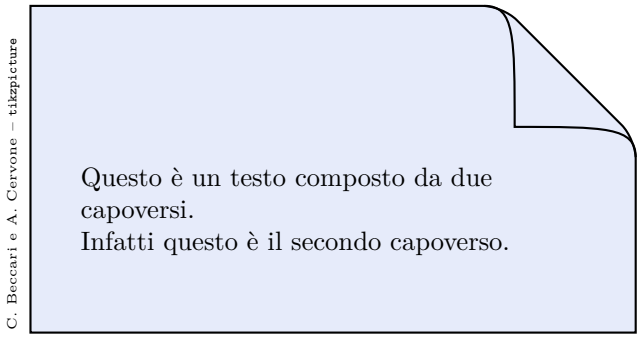


Figura 9.8: Una paginetta con l'angolo ripiegato

Nell'esempio della figura manca il rientro del capoverso; è voluto anche per mostrare quanto sia comodo quel rientro per capire subito che si sta leggendo un nuovo capoverso.

9.5 I vantaggi di usare programmi nativi del sistema TEX

I vantaggi di usare i programmi nativi del sistema TEX (*pgf* funziona anche con Plain TEX e con ConTEXt) consistono nel fatto che questi sistemi consentono di usare direttamente gli stessi font che vengono usati per il testo. Non si trascuri questa possibilità, perché l'uso di font diversi, talvolta incompatibili o non adatti a quello che si sta scrivendo, è un difetto così grave che salta all'occhio anche della persona più inesperta.

I disegni, per esempio, prodotti con famosi programmi interattivi di matematica, possono essere salvati in formato PostScript; tuttavia è poi necessario correggere questi file per poter cambiare i font, specialmente quando alcune scritte che compaiono nei disegni sono costituite da formule e chiunque vede subito che non sono composte con la stessa maestria di quelle composte con LATEX. Esistono dei pacchetti, per esempio *psfrag*, che riescono ad eseguire alcune di queste trasformazioni, ma resta sempre un lavoro maggiore da fare che richiede una attenzione maggiore, come succede sempre quando si eseguono delle correzioni.

Non si trascuri, quindi, la possibilità di disegnare semplici diagrammi ricorrendo anche al più elementare ambiente da disegno costituito da *picture*. Il vantaggio dell'uso dei font giusti supera qualunque fatica si debba fare per ottenere un risultato grafico accettabile. Se poi si afffronta la difficoltà di imparare ad usare l'ambiente *tikzpicture*, allora non si deve rinunciare proprio a niente.

9.6 METAPOST

Qui si vuol dare un ulteriore suggerimento al lettore interessato: egli può ricorrere al programma METAPOST per eseguire disegni al tratto sia bidimensionali sia tridimensionali.

Il programma METAPOST è descritto abbastanza dettagliatamente nel documento `.../doc/metapost/base/mpman.pdf`; è utile consultare anche il documento `.../doc/metapost/base/mpintro.pdf`; in ogni caso METAPOST prevede la conoscenza di METAFONT, quindi tutta l'operazione di documentazione ed apprendimento può sembrare piuttosto complessa. A questo proposito vale la pena di rompere il ghiaccio con `.../doc/english/metafp/metafp.pdf`, che è un testo introduttivo di Peter Wilson per iniziare ad usare sia METAFONT, sia METAPOST.

L'utilità di usare METAPOST per eseguire i disegni consiste nel fatto che si possono usare i costrutti più complessi che vengono elaborati da quel programma in modo che il file di uscita sia compatibile sia per il driver dvips sia per il linguaggio PDF (in pratica pdflatex, lualatex e xelatex insieme al suo convertitore xdvipdfmx), cosicché il disegno prodotto può essere utilizzato sia per comporre un documento in formato PostScript sia in formato PDF senza bisogno delle successive trasformazioni da un formato all'altro. Si veda anche il paragrafo 10.3.

Un altro vantaggio dell'uso di METAPOST consiste nel fatto che tutte le indicazioni testuali e matematiche da inserire nel disegno vengono composte dal programma di composizione stesso grazie ai comandi contenuti nel file sorgente METAPOST; l'uso dei font compatibili con quelli del testo composto è quindi assolutamente garantito.

A titolo di esempio si riporta il listato METAPOST del disegno che compare nella figura 9.9. I punti indicati con $\mathtt{z}_i$ sono alcune coordinate del piano che rappresentano i punti di appoggio per circonferenze e segmenti; le grandezze `pout` e `pin` sono dei *path*, cioè dei percorsi definiti tramite le precedenti coordinate e i comandi che li raccordano con curve di Bézier di terzo grado con concavità modeste (sequenze di tre puntini); i comandi `fill` e `draw` fanno esattamente quello che dice il loro nome. Interessanti sono i comandi per etichettare i punti: `dotlabel.top` disegna un punto nero nel punto `z0` e lo etichetta con quanto scritto fra parentesi. Invece `label.top` non disegna niente, ma mette l'etichetta contenuta fra parentesi a metà strada fra il punti `z0` e `z3`. Entrambi questi comandi per etichettare scrivono in linguaggio TeX quanto è contenuto fra `btex` ed `etex`; vi si leggono infatti le specificazioni per due oggetti matematici (da comporre con lo stile `\textstyle`); uno è il nome del centro del cerchio esterno O, e l'altro è l'indicazione del raggio r.

```
beginfig(1);
u:=1mm;
path pout,pin;
z1=(0,20u);
z2=(20u,40u);
```

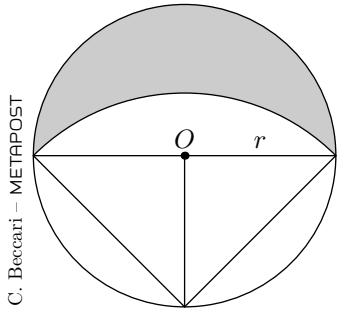


Figura 9.9: Un semplice esempio di figura geometrica composta con METAPOST

```
z3=(40u,20u);
z4=(20u,20*sqrt(2)*u);
z0=.5[z1,z3];
pout= z1{up}...z2{right}...{down}z3;
pin= z3{z2-z3}...z4{left}...{z1-z2}z1;
fill pout&pin&cycle withcolor .8white;
z5=(20u,0);
draw pout...z5{left}...cycle;
draw pin;
draw z1--z3;
draw z0--z5;
draw z5--z1;
draw z5--z3;
dotlabel.top(btex$O$etex,z0);
label.top(btex$r$etex,.5[z0,z3]);
endfig;
end
```

Se il breve listato precedente è contenuto nel file `cerchi.mp` e, da linea di comando, si dà il comando[5]

```
mp cerchi
```

si ottiene il file `cerchi.001`, a cui si può cambiare nome in `cerchi.mps`, per poi importarlo nel documento da comporre con il comando `\includegraphics` di cui si parlerà nel prossimo capitolo[6].

[5] Il comando per lanciare METAPOST può essere `mp` oppure `mpost` a seconda dell'installazione del sistema TEX.

[6] Alternativamente si può usare il comando METAPOST `outputtemplate` descritto nella pagina 260.

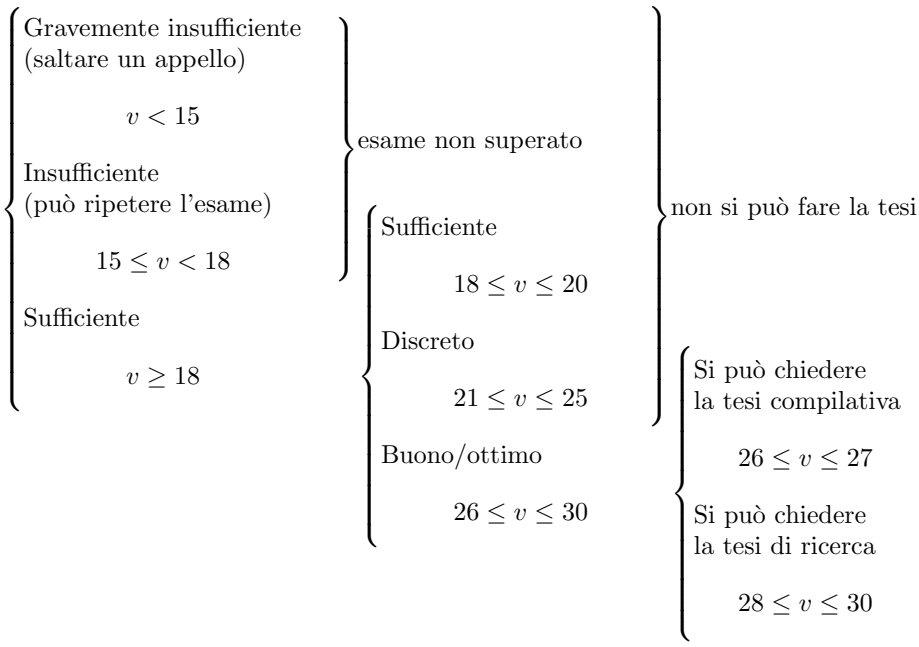


Tabella 9.1: Tabella insolita

9.7 Usi insoliti dell'ambiente *picture*

Le tabelle, come si è detto alla fine del capitolo 8, possono presentare dei problemi e si è indicato come risolverli, almeno in certi casi; la tabella 9.1 contiene sicuramente del materiale ordinato in celle giustapposte, ma le grandi parentesi graffe certamente disturbano gli allineamenti, perché si protraggono oltre i limiti delle celle che intendono raggruppare nelle varie colonne.

Se si compone la tabella nell'ambiente *picture* invece che nell'ambiente *tabular* bisogna rinunciare agli automatismi che quest'ultimo ambiente mette a disposizione, ma l'ambiente *picture* consente di collocare con precisione qualunque oggetto fornendo le coordinate del suo punto di riferimento, ma anche nascondendone le dimensioni reali.

Perciò con l'ambiente *picture* si possono parzialmente sovrapporre oggetti, pur di avere la pazienza di determinare le posizioni dei punti di collocamento con alcune prove. La tabella 9.1[7] è stata composta con il codice seguente, dove le coordinate da dare come argomento ai vari comandi `\put` sono state determinate sperimentalmente.

```
\newcommand*\rBrace[1]{%
                \makebox(0,0)[r]{$\left.\rule{0pt}{#1}\right\}$}}
```

[7] L'esempio è stato suggerito da Enrico Bellino, Università Cattolica del Sacro Cuore.

```
...
\setlength{\unitlength}{1ex}
\begin{picture}(85,60)
%
\put(0,40){$\left\{\parbox{25\unitlength}{\null
  Gravemente insufficiente\newline (saltare un appello) \[v<15\]
  Insufficiente\newline (può ripetere l'esame)\[ 15 \leq v<18\]
  Sufficiente \[ v\geq 18\]}\right.$}
%
\put(32,46){\rBrace{4.5\baselineskip}esame non superato}
%
\put(32,25){$\left\{\parbox{25\unitlength}{\null
  Sufficiente \[18\leq v\leq 20 \]
  Discreto \[ 21\leq v\leq 25\]
  Buono/ottimo \[26\leq v\leq 30\]}\right.$}
%
\put(60,40){\rBrace{7\baselineskip}non si può fare la tesi}
%
\put(60,14.5){$\left\{\parbox{20\unitlength}{\null
  Si può chiedere\newline la tesi compilativa \[26\leq v\leq 27\]
  Si può chiedere\newline la tesi di ricerca \[28\leq v\leq 30\]}
\right.$}
%
\end{picture}
```

La definizione che precede i tre puntini è nel preambolo; essa definisce il comando `\rBrace` che serve per comporre una parentesi graffa di semialtezza specificata, con il punto di riferimento al suo centro, ma racchiusa dentro una scatola di dimensioni nulle; in questo modo la parentesi graffa può essere collocata ovunque senza che causi nessun intralcio al posizionamento degli altri oggetti: è come se non ci fosse.

Successivamente, prima di aprire l'ambiente *picture* (ma dentro all'ambiente *table* per rendere il tutto flottante e con il titolino della didascalia corretto) si assegna all'unità di misura `\unitlength` un valore dipendente dal font in uso: 1 ex.

Aperto l'ambiente *picture* si collocano i vari oggetti; si notino le scatole definite mediante `\parbox` che consentono di comporvi dentro del testo come se si trovasse dentro una colonna con la giustezza pari a quanto specificato nel primo argomento. Inoltre il primo elemento che vi si compone è la scatola nulla `\null`, la quale impegna una riga, del tutto bianca; serve per bilanciare lo spazio che si trova in fondo alla scatola dopo l'equazione centrata. Il resto di quanto si trova a formare il secondo argomento di `\parbox` è codice senza sorprese.

Si nota invece che le grandi parentesi graffe chiuse, costruite mediante il comando `\rBrace` e la semialtezza specificata, non occupando nessuno spazio, non alterano il collocamento del testo dell'unica riga che le accompagna.

La tabella 9.1 non è il massimo dell'eleganza, ma talvolta è necessario costruire tabelle di questo genere usate abitualmente in certe discipline scientifiche. In queste circostanze ricorrere all'ambiente *picture* sembra essere la soluzione più semplice.

9.8 Linee guida per la grafica

Per creare della grafica ben fatta bisogna avere talento e seguire alcune linee guida; con il permesso dell'autore, Till Tantau, qui si riproduce il capitolo 4 della prima parte del suo manuale [62] corredato con alcuni commenti messi in evidenza con il font di questo capoverso, in modo che si stacchino bene dal testo originale, sia pure tradotto con un certa libertà. In una revisione successiva della documentazione quanto qui tradotto corrisponde pressappoco al capitolo 7 prima parte della nuova versione.

Si noti che nel seguito con la parola *grafica* si intende sia l'attività connessa con la produzione di disegni, sia il prodotto di questa attività, cioè uno o più disegni. Con *grafici* si intendono i disegni in senso stretto, generalmente disegni al tratto, quasi sempre privi di sfumature di colore, e che possono avere caratteristiche molto diverse.

I *diagrammi* sono disegni dell'andamento di una o più curve riferite ad assi cartesiani nel piano, meno frequentemente sono la proiezione sulla carta di diagrammi tridimensionali. Gli *schemi a blocchi* o i *diagrammi di flusso* sono disegni di scatole rettangolari o altre forme geometriche chiuse, spesso con sfondi colorati, collegati fra di loro da frecce che descrivono il fluire delle informazioni da una scatola all'altra. Gli *istogrammi* descrivono in due dimensioni, spesso con effetti tridimensionali, distribuzioni statistiche assai spesso non collegate ad una variabile indipendente misurabile. I *diagrammi a torta*, analogamente, descrivono delle distribuzioni statistiche mediante la lunghezza di archi di una circonferenza o l'area di settori di un cerchio. I *cartogrammi* rappresentano distribuzioni proiettate sul piano di terreni o di aree geografiche distinte in modo che sia agevole risalire sia al loro perimetro, sia alla loro superficie, e includono le carte geografiche in senso stretto, le mappe e le planimetrie. I *pittogrammi* possono avere diversi aspetti e rappresentare le cose più diverse mediante aggregazioni di disegnini che rappresentano cose, animali o persone. I *disegni al tratto* più generali, non classificabili come descritto sopra, possono essere delle rappresentazioni schematiche o prospettiche della realtà, dove il disegno, privo di molta dell'informazione che potrebbe essere presente in una fotografia, permette di mettere in evidenza dei particolari, anche in trasparenza, che non sarebbero altrimenti evidenziabili.

In Italia la produzione di questi vari tipi di disegni è regolata dalle norme UNI [64]. Ciò non toglie che la scelta di un dato tipo di rappresentazione non possa essere sottoposta ad ulteriori regole di carattere estetico o di qualità della comunicazione.

9.8.1 Preliminari

Questo paragrafo non riguarda in senso stretto né PGF né Ti*k*Z, ma riguarda le linee guida generali e i principi che presiedono alla grafica per le presentazioni scientifiche, per gli articoli e per i libri.

Le linee guida in questo paragrafo arrivano da diverse fonti. Molte io le chiamerei semplicemente "buon senso", alcune riflettono la mia esperienza personale (sebbene, spero, non le mie preferenze personali), alcune arrivano da libri di tipografia e di disegno grafico, di cui purtroppo non ho composto la bibliografia. La fonte più importante è costituita dai libri di Edward Tufte. Sebbene io non sia d'accordo con ogni cosa scritta in questi libri, molte delle spiegazioni di Tufte sono così convincenti che ho deciso di riportarle fra queste linee guida [63].

La prima cosa che bisognerebbe domandarsi quando qualcuno propone una serie di linee guida è: "Dovrei davvero seguire queste linee guida?" Questa è una domanda importante, perché esistono dei buoni motivi per non seguire linee guida di carattere generale.

- La persona che ha scritto quelle linee guida potrebbe aver avuto degli obbiettivi diversi dai vostri. Per esempio una linea guida potrebbe affermare "usa il colore rosso per dare enfasi". Mentre questa linea guida potrebbe avere senso, per esempio, quando si predispone una presentazione da proiettare, il "colore" rosso ha un effetto *opposto* all' "enfasi" quando viene stampato con una stampante in bianco e nero.

  Le linee guida sono state quasi sempre scritte per affrontare una situazione specifica. Se non siete in quella situazione, seguire una linea guida può fare più male che bene.

- La regola base della tipografia dice che: "Ogni regola può essere infranta se siete *consapevoli* di ciò che state facendo". Questa regola vale anche per la grafica. Detto in modo equivalente, la regola base dice che: "I soli errori che si possono fare in tipografia sono le cose fatte per ignoranza."

  Quando conoscete una regola e quando ritenete che infrangere la regola produca l'effetto desiderato, allora infrangete la regola.

Perciò prima di seguire una linea guida o di decidere di non seguirla, fatevi le seguenti domande:

1. Questa linea guida riguarda davvero la mia situazione?

2. Se faccio l'opposto di quello che dice la linea guida, i vantaggi saranno superiori agli inconvenienti che la linea guida si propone di evitare?

9.8.2 Programmazione del tempo necessario per la creazione della grafica

Quando create un documento che contiene molta grafica, il tempo necessario per creare questa grafica diventa un fattore importante. Quanto tempo bisognerebbe stimare per la creazione della grafica?

Come regola generale supponete che la grafica vi richieda altrettanto tempo di quello necessario per produrre un testo della stessa lunghezza. Per esempio, quando io compongo un documento, impiego circa un'ora per ogni pagina della prima bozza. Poi per le revisioni impiego da due a quattro ore per ogni pagina. Perciò mi aspetto di avere bisogno di circa mezz'ora per creare la prima bozza di un grafico che occupi mezza pagina. Successivamente penso di avere bisogno ancora di una o due ore prima che il grafico sia a posto.

In molte pubblicazioni, anche in ottime riviste, gli autori e i revisori editoriali hanno evidentemente investito molto tempo sul testo, ma sembra che abbiano speso cinque minuti per creare tutta la grafica. I disegni sembra che siano stati aggiunti in un secondo tempo senza un motivo particolare; oppure sembrano una copia della schermata di qualunque cosa il software di statistica dell'autore gli mostrasse. Come verrà spiegato più avanti, la grafica prodotta da programmi come gnuplot con le impostazioni di default sono di qualità assai modesta.

Creare grafici che convogliano in modo corretto la loro informazione al lettore e che nello stesso tempo sono coerenti con il testo è un processo lento e difficile.

- Trattate i disegni come elementi di prima categoria. Essi meritano lo stesso tempo e la stessa energia che richiede il testo.
- Presumibilmente la creazione di grafici richiede *anche maggior tempo* di quello richiesto dalla composizione del testo, perché alla grafica si darà più importanza in quanto i disegni verranno osservati per primi.
- Pianificate di usare lo stesso tempo per la creazione e la revisione dei vostri grafici quanto ne dedichereste ad un testo della stessa estensione.
- Grafici difficili con un'alta densità di informazione vi possono richiedere anche più tempo.
- Grafici molto semplici vi possono richiedere meno tempo, ma verosimilmente voi non volete scrivere un documento con "grafici molto semplici"; per lo meno non lo volete come non vorreste scrivere "testi molto semplici" della stessa estensione.

9.8.3 Piano di lavoro per creare un grafico

Quando scrivete un articolo (scientifico), probabilmente seguite questo schema: avete dei dati o dei risultati da comunicare. La creazione dell'articolo tipicamente comincia con la predisposizione di una scaletta. Poi le varie sezioni vengono riempite con il testo per creare la prima bozza. Questa bozza subisce diverse

revisioni e, spesso dopo revisioni sostanziali, finalmente diventa l'articolo finale. Tipicamente in un buon articolo per una rivista non c'è una sola frase che arrivi alla fine senza aver subito qualche modifica.

Creare la grafica segue lo stesso percorso:

- Decidete dapprima che cosa il grafico deve comunicare al lettore; prendete consapevolmente questa decisione, cioè domandatevi: "Questo grafico che cosa deve dire al lettore?"
- Preparate uno schema grossolano del grafico con la sua forma e i suoi punti essenziali; spesso va bene farlo con carta e matita. Spesso è utile abbozzare il disegno su carta millimetrata o almeno quadrettata.
- Inserite i successivi dettagli per produrre la prima bozza.
- Revisionate il grafico via via che rivedete il testo.

9.8.4 Collegamento fra testo e grafico

I grafici possono essere collocati in diverse posizioni rispetto al testo, nel senso che essi possono venire collocati in linea col testo, ovvero "in mezzo al testo", oppure essi possono essere usati come "figure" autonome. Siccome agli stampatori piace avere le pagine "piene", (per motivi che sono allo stesso tempo economici ed estetici), le figure autonome tradizionalmente possono venire collocate in posizioni relativamente distanti dal testo che vi fa riferimento. LaTeX sembra eseguire questo "allontanamento" per motivi tecnici, cioè per dividere automaticamente il testo in pagine, evitando pagine mozze e/o pagine con i capoversi malamente ed eccessivamente spaziati.

Quando un grafico è inserito in linea col testo, esso viene automaticamente collegato ad esso nel senso che le "etichette" del grafico vengono spiegate e descritte dal testo circostante. E il testo spiega chiaramente ciò a cui il grafico si riferisce e che cosa vi è mostrato.

All'opposto una figura autonoma spesso viene esaminata quando il testo che vi si riferisce o non è ancora stato letto, oppure è stato letto diverse pagine prima. Per questo motivo è bene seguire le seguenti linee guida quando si creano figure autonome o flottanti;

- Le figure flottanti devono avere una didascalia che permetta di renderle "autocomprensibili".

  Per esempio, supponete che un diagramma rappresenti gli stadi successivi dell'algoritmo *quicksort*. Allora la didascalia, come minimo dovrebbe informare il lettore che "La figura mostra gli stadi successivi dell'algoritmo *quicksort* spiegato nella pagina xyz." e non semplicemente "Algoritmo *quicksort*."
- Una buona didascalia aggiunge tanta informazione contestuale quanto è possibile. Per esempio, essa potrebbe specificare: "La figura mostra i

diversi stadi dell'algoritmo *quicksort* spiegato nella pagina xyz. Nella prima linea l'elemento 5 è scelto come cerniera. Questo produce... " Anche se questa informazione può venire inserita nel testo, collocarla nella didascalia assicura che venga mantenuta la contestualità. Non abbiate paura di una didascalia di cinque righe. (Il vostro revisore editoriale vi odierà. Prendete in considerazione la possibilità di ricambiarlo.) Inoltre ricordatevi che la didascalia è composta di tre parti: il titolino corrente, il titolo e, opzionalmente, il testo; sfruttate la possibilità di usare questo testo; nello stesso tempo siate succinti per evitare che LaTeX incontri troppe difficoltà per far flottare la vostra figura in una posizione conveniente.

- Fate riferimento al grafico nel testo con una frase del tipo: "Per un esempio di *quicksort* "al lavoro" vedete la figura m.n nella pagina xyz."

- La maggior parte dei libri di tipografia raccomanda di non usare abbreviazioni del tipo "fig. m.n" ma di scrivere "figura m.n".

  L'argomentazione principale contro le abbreviazioni è che "un punto vale troppo per sprecarlo per una abbreviazione". L'idea è che un punto può indurre il lettore a credere che la frase finisca subito dopo "fig" e che quindi gli sia necessario ritornare indietro coscientemente per rendersi conto che la frase non è affatto finita.

  L'argomentazione a favore delle abbreviazioni è che si risparmia spazio.

  Personalmente nessuna delle due argomentazioni mi convince. Da un lato non ho nessuna prova convincente che le abbreviazioni rallentino la lettura. Dall'altro anche se si abbreviano tutte le parole "figura" in "fig.", è veramente difficile che si risparmi anche una sola riga di testo.

  In alcuni libri di tipografia italiana si trova la raccomandazione di non usare abbreviazioni nel testo principale, ma se ne consente l'uso solo negli incisi, per esempio in quelli fra parentesi.

  Io evito le abbreviazioni.

9.8.5 Coerenza fra testo e grafica

Probabilmente "l'errore" che la maggior parte della gente fa quando crea dei grafici (ricordate che un "errore" nel disegno è sempre semplicemente "ignoranza") è quello di avere un disadattamento fra l'aspetto del grafico e quello del testo.

È abbastanza comune che gli autori usino diversi differenti programmi per creare i loro grafici per un articolo. Un autore potrebbe produrre alcuni diagrammi usando `gnuplot`, alcuni altri usando `xfig` e potrebbe includere alcuni grafici in formato `.eps` ottenuti da qualche collaboratore che ha usato programmi che gli sono ignoti. Tutti questi grafici, verosimilmente, useranno diversi spessori delle linee, font differenti, e avranno dimensioni diverse. Inoltre gli autori spesso usano opzioni come `height=5cm` quando includono dei grafici al fine di scalarli in modo accettabile.

Se si usasse il medesimo approccio per comporre il testo, ogni paragrafo verrebbe composto con un font diverso e di corpo diverso. In un paragrafo i teoremi verrebbero sottolineati, in un altro verrebbero composti in maiuscoletto, in un altro evidenziati in rosso. Inoltre i margini sarebbero diversi in ogni pagina.[8]

Né i lettori né i revisori editoriali tollererebbero un testo scritto in questo modo, ma con la grafica spesso sono costretti a farlo.

Per mantenere una consistenza fra testo e grafica seguite le linee guida seguenti:

- Non cambiate scala ai grafici; in altri termini non riproduceteli a dimensioni diverse da quelle con cui i diagrammi sono stati disegnati.

  Questo significa che quando usate programmi esterni per creare grafici, li dovete creare fin dall'inizio "delle dimensioni giuste".

- Usate gli stessi font per i grafici e per il testo.

- Usate lo stesso spessore delle linee per i grafici e per il testo.

  Lo "spessore delle linee" per il testo normale è in buona sostanza lo spessore delle aste di lettere come la T. Per i font del sistema TEX questo significa 0,4 pt. Tuttavia ci sono alcune riviste che non accettano grafici che contengano linee di spessore inferiore a 0,5 pt.

- Se usate i colori, usateli coerentemente seguendo un "codice dei colori". Per esempio, se usate il rosso nel testo per richiamare l'attenzione del lettore, usate lo stesso colore rosso per richiamare l'attenzione su parti importanti del disegno. Se il colore blu è usato per evidenziare le parti strutturali del testo, come per esempio i titoli dei paragrafi, usate lo stesso colore blu per evidenziare le parti strutturali dei vostri grafici.

  Tuttavia nella grafica si possono usare colori corrispondenti ad una codifica logica e "intrinseca". Per esempio, indipendentemente dal codice dei colori che voi possiate usare, i lettori generalmente associano il colore verde a concetti del tipo "positivo, va bene, OK", mentre associano il colore rosso a concetti come "attenzione, pericolo, azione".

Mantenere la coerenza quando si usano diversi programmi di disegno è quasi impossibile. Per questo motivo dovreste prendere in considerazione di usare sempre lo stesso programma.

9.8.6 Legende nei grafici

Praticamente tutti i disegni contengono delle etichette per individuare parti del grafico o legende testuali che spiegano parti del grafico. Quando usate queste legende, rimanete aderenti alle linee guida seguenti:

[8] *Bisogna convenire che non è raro vedere testi composti in modo così sciatto, quando sono stati usati i soliti word processor; Con LATEX è difficile che ciò accada, non impossibile, ma bisogna mettercela tutta...*

- Siate coerenti nel predisporre queste legende. Lo potete fare in due modi: primo, siate coerenti con il testo principale usando gli stessi font. Secondo, mantenete la coerenza fra una legenda e l'altra, cioè, se componete alcune legende in un certo modo, componete tutte le legende nello stesso modo.

- Oltre ad usare gli stessi font nel testo principale e nel grafico, dovreste anche usare le stesse notazioni. Per esempio, se scrivete 1/2 nel testo principale, usate la stessa scrittura nelle legende, non scrivete 0,5. Un π è un π e non 3,141. Ancora, $e^{-i\pi}$ è e resta $e^{-i\pi}$, non "-1" e meno che mai "-1".

- Le legende devono essere leggibili[9]. Esse pertanto devono essere abbastanza grandi e non devono essere disturbate dalla presenza di linee o di altro testo a loro in parte sovrapposto. Questo, ovviamente, vale anche per le linee e per il testo "dietro" le legende.

- Le legende devono stare "nel posto giusto". Quando c'è abbastanza spazio, le legende dovrebbero essere collocate vicino all'oggetto a cui si riferiscono. Solo se è veramente necessario si può usare una leggera linea che connetta la legenda con l'oggetto. Evitate di usare legende che si riferiscono a spiegazioni contenute in legende esterne, altrimenti il lettore sarebbe costretto a saltare avanti e indietro fra la spiegazione e l'oggetto che viene descritto.

- Prendete in considerazione la possibilità di ridurre l'evidenza di alcune etichette meno importanti affinché il lettore possa focalizzare meglio la sua attenzione sul grafico.

9.8.7 Diagrammi di vario genere

Specialmente negli articoli scientifici, i grafici più frequenti sono i *diagrammi*. Possono essere composti in molte differenti maniere, che vanno dai semplici diagrammi con una sola curva, ai diagrammi parametrici, a quelli tridimensionali, ai diagrammi a torta, e molti altri.

È purtroppo ben noto che è difficile fare dei bei diagrammi. In parte bisogna incolpare le impostazioni di default di programmi come gnuplot o Excel, che consentono di disegnare facilmente brutti diagrammi.

La prima cosa che dovreste domandarvi quando volete creare un diagramma è la seguente:

- Ci sono abbastanza dati da riportare sul diagramma?

Se la risposta è "Per la verità, no", componete una tabella invece di un diagramma.

Una tipica situazione nella quale un diagramma non è necessario è quando si riportano pochi dati su un istogramma a barre. Ecco un esempio reale: Alla fine

[9] Legenda è una parola latina neutra plurale e significa "le cose che devono essere lette".

Valutazione	*Partecipanti (su 50) che diedero questa valutazione*	*Percentuale*
ottimo	3	6%
buono	9	18%
passabile	10	20%
scadente	8	16%
pessimo	0	0%
nessuna	20	40%

Tabella 9.2: Esempio di valutazione di un seminario

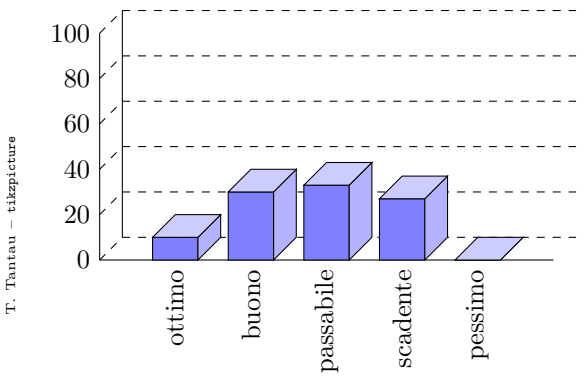


Figura 9.10: Esempio di istogramma per presentare il risultato della valutazione di un seminario

di un seminario l'oratore chiese ai presenti una valutazione. Dei 50 partecipanti, 30 restituirono il questionario di valutazione. La valutazione risultò essere la seguente: tre partecipanti dissero che il seminario era stato ottimo, nove lo considerarono buono, dieci lo considerarono passabile, otto scadente, mentre nessuno lo considerò pessimo.

Un modo semplice per raccogliere questa informazione è rappresentato dalla tabella 9.2.

Quello che, invece, l'oratore fece, fu di presentare il risultato sotto forma di un istogramma a barre, come mostrato nella figura 9.10.

Si osservi che la tabella 9.2 e la figura 9.10 occupano pressappoco lo stesso spazio. Se il vostro primo pensiero è che "il grafico è più gradevole della tabella", provate a rispondere alle seguenti domande basandovi sull'informazione contenuta nel grafico o nella tabella.

1. Quanti partecipanti c'erano?

2. Quanti partecipanti restituirono il questionario?

3. In che percentuale i partecipanti restituirono il questionario?

4. Quanti partecipanti risposero “ottimo”?

5. Fra tutti i partecipanti, quale percentuale rispose “ottimo”?

6. Furono più di un quarto coloro che risposero “scadente” o “pessimo”?

7. Fra coloro che restituirono il questionario, quale fu la percentuale di coloro che valutarono il seminario “ottimo”?

Purtroppo l'istogramma non permette di rispondere a *nemmeno una di queste domande*. La tabella risponde direttamente a tutte le domande, tranne l'ultima, per la quale, inoltre, dà gli elementi per eseguire il calcolo. In sostanza la densità di informazione dell'istogramma è quasi nulla; la tabella ha una densità di informazione molto maggiore, anche se contiene una grossa parte di spazio bianco e pochi numeri.

Ecco che cosa è andato storto con l'istogramma:

- L'intero grafico è dominato dalla presenza fastidiosa delle linee di sfondo.
- Non è chiaro il significato dei numeri a sinistra; probabilmente si tratta di percentuali, ma potrebbe anche essere il numero assoluto dei partecipanti.
- Le legende in basso sono ruotate, così che diventano più difficili da leggere.

  (Nella presentazione reale che io ho visto, il testo era composto con caratteri a matrici di punti campionati su una griglia di non più di 10 pixel per 6 pixel per ogni lettera, con accostamenti errati al punto da risultare quasi illeggibile.)
- La terza dimensione aggiunge al diagramma una certa complessità ma non aggiunge nessuna informazione.
- L'impostazione tridimensionale rende molto più difficile da leggere l'altezza delle barre. Esaminate la barra “scadente”. La sua altezza è maggiore o minore di 20? Mentre la parte frontale della barra è sotto al livello 20, il retro della barra, che è quello che conta, è sopra a tale livello.
- È impossibile dire quali numeri siano rappresentati dalle barre. Perciò le barre nascondono senza necessità proprio l'informazione che dovrebbero dare.
- La somma delle altezze delle barre è 100% o 60%?
- L'altezza della barra “pessimo” è zero o uno?
- Perché le barre sono blu?

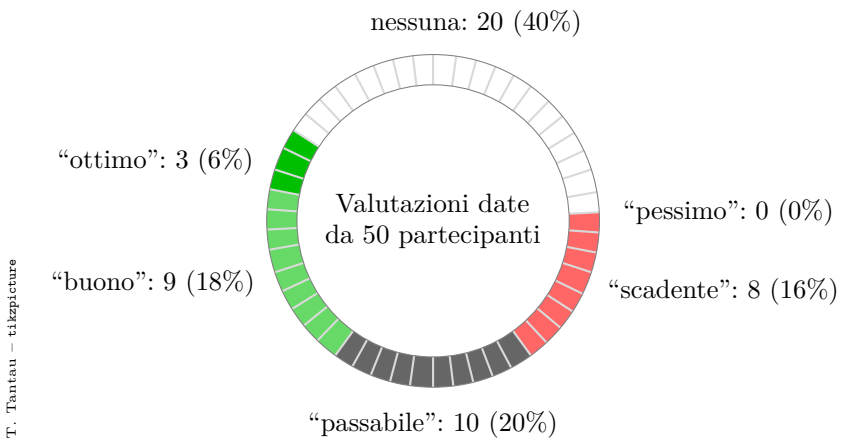


Figura 9.11: Altro esempio di diagramma per presentare il risultato della valutazione di un seminario

Potreste argomentare che in questo esempio i valori precisi non sono importanti per il grafico. La cosa importante è il “messaggio”, cioè che ci sono più valutazioni “ottimo” e “buono” che valutazioni “scadente” o “pessimo”. Ma per trasmettere questo messaggio o si usa una frase, oppure si disegna un diagramma che convogli questo messaggio in modo più chiaro, come nella figura 9.11.

Il grafico della figura 9.11 ha pressappoco la stessa densità di informazione della tabella (ha le stesse dimensioni e vi sono riportati gli stessi numeri). In più uno può direttamente “vedere” che ci sono più valutazioni “buone” e “ottime” che non “scadenti”. Uno può anche “vedere” che il numero delle persone che non hanno formulato valutazioni di nessun tipo, non restituendo il questionario, non è trascurabile, e questo è un fatto abbastanza comune quando si distribuiscono questionari di valutazione.

Anche i diagrammi spaziali spesso non sono efficaci. Esaminiamo un altro esempio nella figura 9.12 che io ho ridisegnato traendolo dalla rivista *Die Zeit*, 4 giugno 2005. Questo diagramma della figura 9.12 è stato ridisegnato con Ti*k*Z, ma è praticamente identico all’originale.

A prima vista esso sembra ben fatto, completo e pieno di informazioni, ma ci sono diverse cose che non sono a posto.

- Il grafico è tridimensionale. Tuttavia le sfumature di colore non aggiungono nulla dal punto di vista informativo, al più distraggono.

- In un grafico 3-D le dimensioni relative sono molto distorte. Per esempio, l’area occupata dalla zona grigia “Braunkhole” è maggiore dell’area occupata dalla zona verde della “Kernenergie”, *malgrado il fatto che la percentuale dell’energia prodotta con il Braunkhole sia inferiore alla percentuale di quella nucleare.*

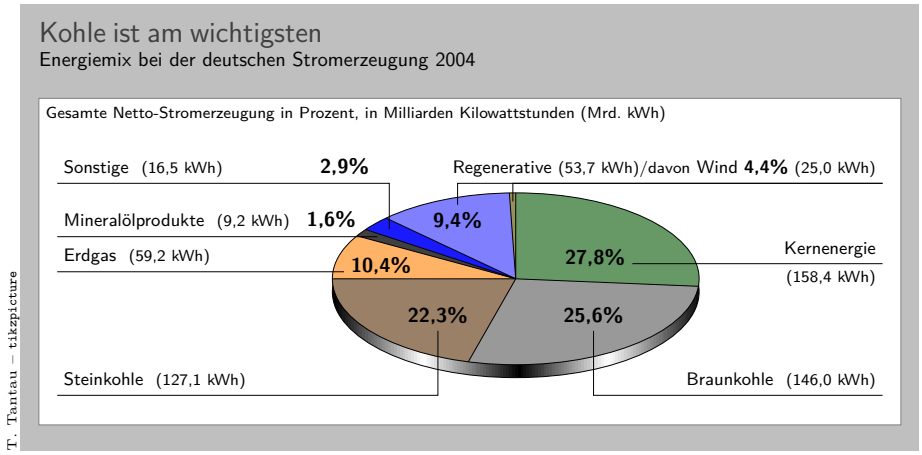


Figura 9.12: Esempio di diagramma a torta per confrontare le fonti di produzione dell'energia elettrica in Germania nel 2004. Regenerative: rinnovabile; Wind: eolica; Kernenergie: nucleare; Braunkhole: lignite; Steinkhole: litantrace; Erdgas: metano naturale; Mineralölprodukte: petrolio e derivati; Sonstige: altre fonti.

- La distorsione della grafica 3-D diventa ancora peggiore per le piccole aree. L'area dell'energia "Regenerative" risulta più grande di quella occupata dall'area destinata all' "Erdgas". L'area del "Wind" è appena minore di quella "Mineralölprodukte" *anche se la percentuale del Wind è quasi tre volte maggiore di quella del Mineralölprodukte.*

  In quest'ultimo caso le dimensioni diverse sono solo in parte dovute alla distorsione 3-D. I disegnatori del grafico originale hanno anche fatto la fetta "Wind" troppo piccola anche tenendo conto della distorsione. (Si confrontino in generale le dimensioni del "Wind" con quelle del "Regenerative".)

- Stando alla didascalia di questo grafico, esso dovrebbe darci l'informazione che in Germania il carbone è stata la sorgente di energia più importante nel 2004. Tralasciando le forti distorsioni prodotte dall'impostazione 3-D, ci vuole della buona volontà per percepire il messaggio.

  Il carbone come fonte di energia è diviso in due fette, una per il "Braunkhole" e l'altra per il "Steinkhole" (due diversi tipi di carbone). Quando si uniscano le due fette, solo allora si vede che la metà inferiore della torta è presa dal carbone.

  Inoltre le due aree per i due differenti tipi di carbone non sono per niente collegate visualmente. Invece sono usati due colori differenti e le rispettive legende sono ai lati opposti del disegno. A confronto, "Regenerative" e "Wind" sono collegate molto strettamente.

- Il codice dei colori del grafico non segue nessuna regola logica. Perché l'energia nucleare è segnata in verde? Perché l'energia rinnovabile è azzurra e quella derivata da altre fonti è blu? Sembra addirittura una barzelletta che l'area del Braunkhole (letteralmente del carbone marrone) sia grigio pietra, mentre l'area dello Steinkhole (letteralmente carbone di pietra) sia marrone.

- L'area con il colore più chiaro è destinata all' "Erdgas". Quest'area emerge dal diagramma con più forza, sebbene secondo questo diagramma l' "Erdgas" non sia veramente importante.

Edward Tufte chiama i grafici come quello della figura 9.12 "grafici spazzatura".

Ecco allora alcune raccomandazioni per aiutarvi a non produrre grafici spazzatura:

- Non fate diagrammi a torta tridimensionali: questi sono *spazzatura.*

- Esaminate se non sia meglio preferire una tabella a un diagramma a torta.

- Non usate i colori a caso; usateli invece per dirigere l'attenzione del lettore e per raggruppare gli oggetti.

- Non usate tessiture di sfondo come le tessiture a graticcio o di linee diagonali; usate piuttosto i colori. Le tessiture distraggono. Le tessiture di sfondo nei diagrammi informativi sono *spazzatura.*[10]

9.8.8 Attenzione e distrazione

Prendete il vostro romanzo preferito ed esaminate una pagina tipica. Noterete che la pagina è molto uniforme. Non c'è nulla che possa distrarre il lettore mentre legge. Non ci sono grandi titoli, non c'è testo composto in neretto, non ci sono spazi bianchi di grandi dimensioni. Invece se l'autore vuole evidenziare qualcosa lo fa componendolo in corsivo. Il corsivo si adatta benissimo al font usato per il testo principale, verosimilmente il tondo, e a una certa distanza non riuscireste a dire se la pagina contiene lettere in corsivo, mentre notereste subito se ci fosse anche una sola una parola composta in nero. Il motivo per il quale i romanzi sono scritti così segue il paradigma seguente: "Evita le distrazioni".

La buona tipografia (come anche la buona organizzazione) è qualcosa che *non* si nota. Il compito della tipografia è quello di rendere il più possibile comoda la lettura del testo e quindi del recepimento dell'informazione che esso contiene. Per un romanzo i lettori recepiscono l'informazione leggendo il testo una riga dopo l'altra, come se stessero ascoltando qualcuno che racconta una storia. In questa

[10] Dovendo stampare in bianco e nero e tonalità di grigio, usate preferibilmente sfondi di diverse tonalità di grigio piuttosto che tessiture di vario genere. Fate attenzione a non scegliere grigi troppo chiari o troppo scuri perché in stampa potrebbero apparire decisamente bianchi o neri. Nello stesso tempo fra grigi adiacenti state attenti ad usare grigi contrastanti o grigi concordanti a seconda della contestualizzazione delle aree da rappresentare.

situazione qualunque cosa ci sia sulla pagina, che distrae l'occhio dall'andare rapidamente e uniformemente da una riga alla successiva, rende la lettura del testo meno agevole.

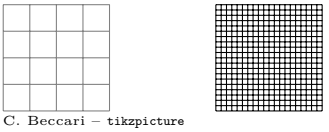
Ora prendete la vostra rivista preferita o il vostro quotidiano ed esaminate una pagina tipica. Noterete che la pagina contiene un gran "movimento". I font sono usati con corpi differenti e con strutture compositive diverse; il testo è organizzato in colonne strette, spessissimo interrotte da immagini. I motivi per i quali le riviste e i quotidiani sono composti in questo modo segue un altro paradigma: "Guida l'attenzione".

I lettori non leggono una rivista come un romanzo. Invece di leggere una rivista una riga dopo l'altra, essi leggono i titoli o dei brevi riassunti (sottotitoli) per verificare se sono interessati a leggere l'intero articolo. Nel momento, però, in cui il lettore ha deciso che vuole leggere l'articolo, egli non vuole più essere distratto ed è per questo che gli articoli sono scritti proprio come i romanzi.

I due principi "evita le distrazioni" e "guida l'attenzione" si applicano anche alla grafica. Quando disegnate un grafico dovete eliminare ogni cosa che "distragga l'occhio". Nello stesso tempo dovete cercare attivamente di guidare il lettore "attraverso il grafico" usando i font, i colori e gli spessori delle linee per evidenziare le varie parti.

Qui c'è una lista, assolutamente non completa, di cose che possono distrarre il lettore:

- I forti contrasti attraggono sempre l'occhio. Per esempio, si considerino le seguenti due griglie:


C. Beccari – tikzpicture

  Sebbene la griglia di sinistra appaia per prima quando noi leggiamo la pagina nella direzione solita da sinistra a destra, è verosimile che la griglia di destra sia percepita per prima; il contrasto bianco-nero è maggiore di quello che si trova nel contrasto grigio-bianco[11]. Inoltre nella griglia di destra ci sono più punti di cambiamento di contrasto che ne aumentano l'effetto.

  Oggetti come le griglie e, in generale, le linee delle coordinate, non dovrebbero di solito attrarre l'attenzione del lettore e quindi dovrebbero essere disegnate con un debole contrasto rispetto allo sfondo. Inoltre una griglia a maglie larghe distrae di meno di una griglia a maglie strette.

[11] Se state esaminando queste due griglie sullo schermo, tenete conto che le linee della griglia di sinistra sono tracciate in colore grigio, il grigio di default con 75% di bianco e 25% di nero; la griglia di destra è disegnata in nero. La qualità dello schermo e la sua tecnologia, nonché la sua densità di pixel potrebbe rendere difficile la percezione di queste differenze.

- Le linee tratteggiate creano molti punti in cui c'è un contrasto bianco-nero. Le linee tratteggiate o punteggiate in generale distraggono molto.

  Non usate linee con differenti tratteggi per differenziare curve diverse. Si perdono dei dati in questo modo (a meno che lo stacco fra i trattini e/o i puntini non sia molto piccolo) e l'occhio non riesce bene a raggruppare gli oggetti sulla base del tratteggio. L'occhio riesce *molto meglio* a raggruppare gli oggetti sulla base dei colori.

  Stampando in bianco e nero talvolta è necessario violare questa raccomandazione: progettando bene i diagrammi è possibile usare i tratteggi, come è possibile usare tonalità di grigio. Tuttavia se le dimensioni e le distanze delle linee da tracciare (mai più di quattro sullo stesso diagramma) lo consentono, si possono tracciare le varie curve con linee continue differenziandole mediante l'apposizione di triangolini, quadratini, cerchietti, eccetera, su ciascuna curva, specialmente se collocati in corrispondenza dei punti (nodi) da tracciare lungo ciascuna curva.

- Inserire degli sfondi con linee diagonali, verticali o orizzontali o anche semplicemente con dei puntini, risulta quasi sempre una distrazione e tipicamente non serve a niente.

- Piccole immagini (clip arts) possono essere simpatiche ma di solito distraggono.

9.8.9 Commenti

Till Tantau con molta modestia afferma che ha preso queste raccomandazioni e queste linee guida da libri famosi e autorevoli, in particolare da quelli di Edward Tufte. Egli spera che queste raccomandazioni non rappresentino le sue personali preferenze, ma siano il meglio che è riuscito a distillare da quei libri.

In alcuni casi le linee guida indicate sembrano un po' troppo cogenti o, benché perfettamente comprensibili, lasciano il dubbio di essere difficili da applicare.

Certo, quello che si deduce da queste righe è che, come la composizione tipografica è un'arte che va appresa e coltivata, così la grafica richiede non poca dedizione e studio oltre alla abilità personale del disegnatore.

Oggi dovrebbe essere più facile di ieri trovare un buon grafico che possa fare i nostri disegni; ma anche il miglior grafico di questo mondo non farà dei bei disegni se noi stessi non siamo capaci di informarlo di che cosa il disegno deve trasmettere al lettore; è molto più facile farlo con lo scritto che con il disegno.

Tuttavia quelli di noi che devono produrre documenti scientifici con un certa dose di grafici, diagrammi e pittogrammi è bene che imparino queste linee guida (compreso il fatto che le linee guida possono anche non

essere rispettate se lo si fa con cognizione di causa). Esistono delle norme UNI [64] che danno altre linee guida (non necessariamente in contrasto con quelle qui esposte, forse solo esposte con parole diverse) che chi deve produrre disegni tecnici dovrebbe sempre osservare. Debbo dire che gli esempi riportati nella versione cartacea di queste norme sono bellissimi e, per quello che si può dire dal risultato visivo, sembrano perfettamente coerenti con le linee guida qui esposte.

È bene sottolineare che le norme di disegno statunitensi sono molto diverse da quelle europee; un disegno fatto seguendo quelle norme si distingue a colpo d'occhio; ma si distingue a colpo d'occhio anche un disegno costruito senza seguire nessuna norma; di solito il risultato non è affatto bello.

Capitolo 10

LaTeX: l'importazione di figure esterne

10.1 Introduzione

Come accennato nel capitolo precedente, capita di dover importare disegni o immagini prodotte con strumenti esterni. Per quel che riguarda i disegni al tratto si è già spiegato quali vantaggi ci siano ad usare i programmi di disegno nativi del sistema TeX. Tuttavia può succedere di dover importare disegni o immagini prodotti all'esterno.

Qui ci concentreremo sulle immagini, sostanzialmente le fotografie o altri disegni a mezze tinte non ottenibili da un semplice programma specializzato, per esempio, nel tracciare diagrammi a due o a tre dimensioni.

Il pacchetto *pgf* gestisce anche l'importazione delle immagini esterne, ma lo strumento principe per operare in questa direzione è senz'altro il pacchetto *graphicx* che da anni costituisce l'interfaccia di riferimento.

Prima però di addentrarci nell'uso di quel pacchetto bisogna fermarsi un attimo per meditare sulla moltitudine di formati grafici che rendono le cose meno semplici di quanto potrebbero essere.

10.2 I formati grafici

I formati grafici si distinguono sostanzialmente in formati vettoriali e in formati a matrici di pixel. Essi hanno caratteristiche diverse e per certe applicazioni vanno meglio i primi, mentre per certe altre vanno meglio i secondi.

10.2.1 I formati vettoriali

I formati vettoriali sono descritti mediante una specie di programma che contiene le istruzioni per tracciare i contorni delle aree dove il colore è sostanzialmente

costante; questi contorni sono costituiti dalla conoscenza delle serie di punti per i quali devono passare le curve di Bézier che li descrivono. Questi formati sono usatissimi per tracciare anche quei particolari disegni costituiti dai caratteri dei vari font; questi infatti devono potersi disegnare rapidamente, ma, specialmente se devono essere presentati sullo schermo, devono essere scalabili a piacere senza che l'ingrandimento o il rimpicciolimento producano difetti visibili a occhio nudo.

È chiaro che se il disegno è costituito dai contorni delle aree dello stesso colore, il problema del cambiamento di scala ha una soluzione banale, in quanto basta moltiplicare le coordinate dei punti che descrivono il contorno per il fattore di scala; per il resto non cambia praticamente nulla. In realtà, quando si rimpicciolisce un disegno che contiene delle linee sottili, queste diventano ancora più sottili e, per esempio, sullo schermo, che ha una densità di puntini luminosi abbastanza bassa, da 70 a 100 puntini al pollice, queste linee diventerebbero troppo sottili per essere rappresentate; l'algoritmo di rappresentazione deve contenere quindi dei suggerimenti (*hints*) per produrre un effetto finale che non lasci sparire le linee troppo sottili.

In questo testo l'argomento diventerebbe troppo tecnico e non si insiste oltre; tuttavia vale la pena di ricordare che i formati grafici vettoriali più noti e frequenti sono il formato PostScript (estensione del file `.ps`) e il suo parente stretto Encapsulated PostScript (estensione `.eps`); anche il formato Portable document Formata (estensione `.pdf`) è vettoriale. Oggi sta diventando sempre più frequente il formato Scalable Vector Graphics (estensione `.svg`), usato specialmente per le applicazioni Web.

C'è anche il formato di uscita del programma METAPOST che, partendo da un file sorgente simile a quanto si potrebbe scrivere con il programma di creazione dei font creato da Donald E. Knuth, produce un'uscita in una specie di linguaggio PostScript ridotto e semplificato. Il programma è molto utile per produrre disegni al tratto, che vanno dai disegni ai diagrammi, ma l'unico difetto è che il linguaggio non è dei più semplici e quindi il programma è usato meno di quanto meriterebbe.

Il formato Portable Document Format (estensione `.pdf`) è un formato vettoriale. Tuttavia il file è un file con il codice compresso (zippato, secondo una diffusa terminologia), ma il file decompresso è sostanzialmente un sotto insieme del formato PostScript, quindi altrettanto vettoriale. Il suo tipo di compressione richiede programmi speciali per la visualizzazione e stampa, destinati specialmente a rendere la pagina in forma forma grafica.

10.2.2 I formati diversi da quelli vettoriali

Le fotografie sono, oggi più spesso di ieri, in formato JPEG (estensione `.jpg`). Tuttavia i formati di matrici di pixel sono diffusissimi, e le estensioni vanno da `.bmp` a `.wmf`, da `.tiff` a `.gif` e a Portable Network Graphics (estensione `.png`); questa è solo una piccola elencazione di formati grafici, perché esistono dei formati specifici per certi particolari apparecchi fotografici digitali o per certi codici di colore.

Il vantaggio dei formati a matrici di pixel è che i programmi per la loro presentazione sono particolarmente rapidi a produrli in forma visibile per gli umani. Tuttavia hanno un paio di difetti tutt'altro che trascurabili.

Il primo difetto è che un'immagine di pochi pixel non può essere ingrandita su un'area che contenga più pixel, perché essa non contiene abbastanza informazione per colorare correttamente i pixel della riproduzione. Un'immagine quadrata di 100 pixel per 100 pixel può essere rappresentata abbastanza bene sullo schermo di un PC ed apparirà un'immagine di circa 30 mm per 30 mm. Se si vuol vedere a pieno schermo la stessa immagine, cioè su un'area di 800 pixel per 640 pixel, si capisce bene che il meglio che si possa fare è di rappresentare i pixel dell'immagine come quadrati di circa 8 pixel per 8 pixel, cioè sgranando l'immagine, cosicché appare come un mosaico di quadrati tutt'altro che piccoli, con la conseguenza che i contorni non sono più nitidi. La situazione peggiora notevolmente sulla carta dove il quadrato di 100 pixel per 100 pixel copre un quadratino di circa 8 mm per 8 mm stampando con una stampante con una densità di 300 puntini al pollice e di circa 4 mm per 4 mm stampando a 600 puntini al pollice; se si volesse ingrandire l'immagine come la si vede sullo schermo, l'effetto mosaico sarebbe talmente importante che difficilmente si riuscirebbe a riconoscere l'immagine.

È vero, esistono programmi che consentono di eseguire un filtraggio bidimensionale in modo da ridurre l'effetto mosaico, ma questo si paga con una minore definizione dei contorni, cosa che spesso rappresenta il male minore, tenuto conto della poca informazione contenuta nel file di partenza; non si pensi di ottenere quei meravigliosi effetti fantascientifici che vengono mostrati in televisione, dove la polizia ingrandisce le immagini della videosorveglianza (che notoriamente riprendono le immagini con una definizione piuttosto bassa) fino a leggere numeri di targa o a riconoscere i lineamenti del colpevole!

La situazione è un poco migliore se si vuole ridurre l'immagine su un'area che contenga un po' meno pixel, ma il tutto è affidato alla qualità del tipo di rappresentazione o di stampa. Conclusione: si preferiscano sempre i formati vettoriali che possono essere ingranditi e rimpiccioliti a piacere. Tuttavia talvolta "bisogna fare di necessità virtù".

Se a questo si aggiunge il secondo inconveniente dovuto al fatto che alcuni formati sono compressi in una maniera che sfrutta la ridondanza delle immagini, ci si rende conto che un simile metodo perde parte dell'informazione e quando il file viene decompresso per la resa dell'immagine su schermo o su carta, appaiono degli artefatti che derivano dall'impossibilità di comprimere il file in modo decisivo senza perdere parte dell'informazione che l'immagine contiene. Se si tratta di una fotografia con colori sfumati e senza contorni netti, questo disturbo si nota poco, ma se si trattasse di un disegno al tratto la resa potrebbe essere inaccettabile.

Per le fotografie il formato `.jpg` va abbastanza bene ma può variare notevolmente da una fotografia all'altra. Per i disegni al tratto, se non fosse possibile avere dei disegni in formato vettoriale, allora il formato più adeguato sarebbe quello con estensione `.png`. Entrambi i formati sono compressi, ma la compressione del formato `.jpg` è una *lossy compression*, cioè con perdita di informazione,

mentre per il formato `.png` si ha una *lossless compression*, cioè senza perdita di informazione. La compressione del formato `.jpg` è più efficace di quella `png` ma a scapito della qualità dell'immagine. Il formato `.png` ha anche un vantaggio rispetto a quello `.jpg`, nel senso che gestisce le trasparenze di colore che, invece, il formato `.jpg` non può gestire.

In ogni caso, fra i formati di questo paragrafo, quello di gran lunga preferibile sarebbe il formato `.pdf`, almeno nelle circostanze in cui il file PDF è stato composto senza perdere la natura vettoriale del disegno di partenza.

Tanto per rendere più visibile il fenomeno appena descritto, la figura 10.1 mostra due diagrammi ottenuti con Excel e salvati uno in formato PNG e l'altro in formato PDF. Per entrambi si è ingrandito di circa 10 volte il quadratino evidenziato sulla parte di sinistra. Come si vede, l'effetto di quantizzazione nella figura in formato PNG rende l'ingrandimento assolutamente irriconoscibile, mentre l'ingrandimento della figura vettoriale in formato PDF è perfetto e si distinguono persino i puntini sui quali Excel ha interpolato la curva.

10.3 I formati accettabili a seconda del programma di composizione

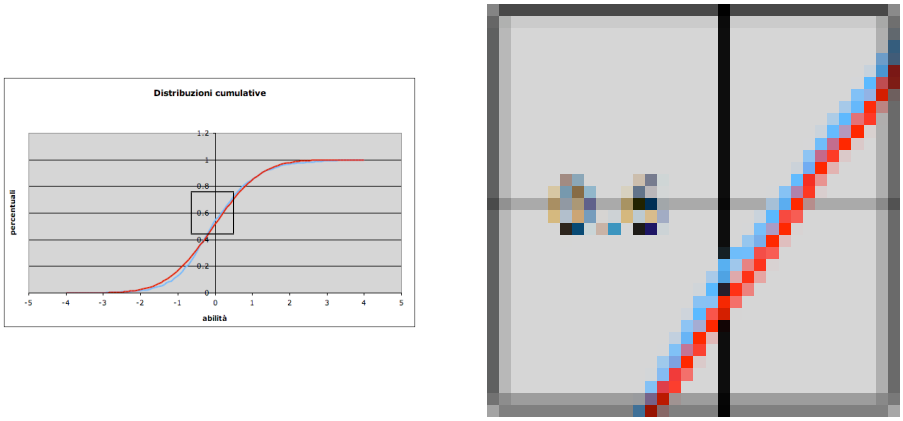
A seconda che si usi latex oppure uno dei programmi pdflatex, lualatex o xelatex, alcuni formati sono accettabili e altri no. Peccato. Il solo formato accettabile da tutti sarebbe il formato di uscita di METAPOST. Ma anche per questo ci vuole un poco di maquillage. L'estensione del nome di questi file è costituita da un numero progressivo, per esempio `.008` perché si trattava dell'ottavo disegno in uscita dall'elaborazione dello stesso file sorgente. Questa estensione è accettabile da pdflatex ma non è accettabile dagli altri programmi. Perciò è necessario cambiare l'estensione di tutti i file di uscita da METAPOST, aggiungendo l'estensione `.mps` o meglio cambiandone il nome, per esempio da `mypicture.008` a `mypicture-008.mps`. Ciò fatto la cosa va bene sia per latex sia per gli altri programmi, ma oltre al modo manuale esiste, dalla versione 1.000 di METAPOST, un modo che sfrutta un meccanismo di denominazione dei file per *modelli*.

Un modello di nome è una stringa racchiusa tra doppi apici che include *codici segnaposto* tra cui `%j` per il nome del file sorgente e `%3c` per il numero di figura con formato fisso a tre caratteri. Otteniamo quindi lo stesso effetto descritto dell'esempio precedente scrivendo l'istruzione di preambolo del file sorgente `.mp`[1]

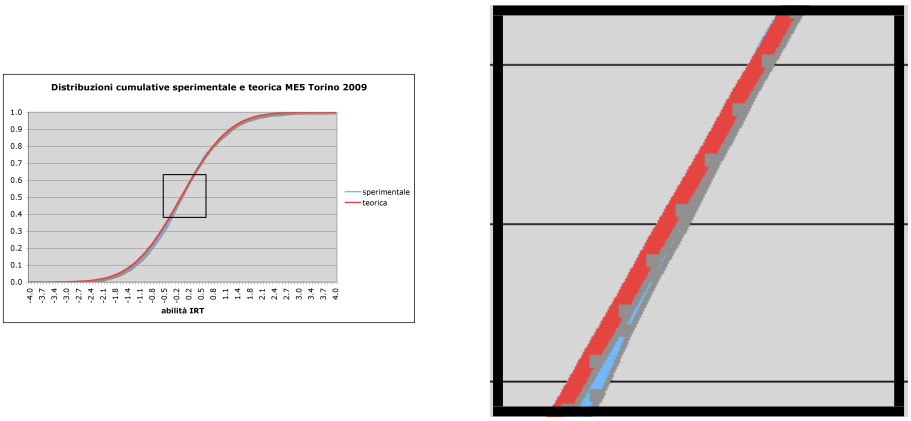
```
outputtemplate := "%j-%3c.mps"
```

Una alternativa legata all'uso del pacchetto *graphicx* potrebbe essere quella di definire una nuova regola di elaborazione dei file grafici; se si usa uno dei program-

[1] Il comando METAPOST `outputtemplate` ha sostituito dalla versione 1.200 il comando `filenametemplate` che quindi oggi è classificata come deprecata. Notizie complete sui codici segnaposto si trovano nel manuale a cui si rimanda.



(*a*)



(*b*)

C.Beccari – picture, Microsoft Excel, Mac OS X Preview

Figura 10.1: Confronto sugli effetti dell'ingrandimento su due figure simili. (*a*) Immagine in formato bitmapped PNG; (*b*) in formato vettoriale PDF; la parte ingrandita a destra corrisponde al riquadro evidenziato nella parte di sinistra.

mi pdflatex o lualatex o xelatex si può usare il comando `\DeclareGraphicsRule` e specificare nel preambolo

```
\DeclareGraphicsRule{*}{mps}{}{}
```

Con questo si dichiara che qualunque altro formato (asterisco), diverso da quelli per i quali esiste una apposita regola, debba venire trattato come un file ottenuto da METAPOST. Questo è un escamotage che presenta qualche inconveniente, ma se si è sicuri di usare solo file ottenuti da METAPOST, non dovrebbe presentarsi nessun problema.

Tuttavia bisogna stare attenti a tutti gli altri formati secondo quanto detto qui di seguito.

10.3.1 I formati accettabili da latex

latex di per sé non accetta nessun formato grafico; sono i driver DVI che accettano questo o quel formato. Il driver dvips accetta solo i formati vettoriali; quindi oltre all'uscita di METAPOST, esso accetta solo i formati `.eps` e `.ps`; il formato Encapsulated è preferibile rispetto al formato PostScript, perché sicuramente non contiene comandi PostScript che risultano incompatibili con l'inserimento delle figure all'interno di un'altra pagina PostScript. Benché il formato possa essere vettoriale, dvips non accetta assolutamente il formato `.pdf`.

Invece il driver dvipdfm accetta sia il formato d'uscita del programma METAPOST, sia i formati vettoriali PostScript `.eps` (preferibilmente) e `.ps`, sia i formati `pdf`, `.jpg` e `.png`. Per questi ultimi necessita di file ausiliari che contengano le dimensioni del rettangolo tangente (bounding box), ma la distribuzione del pacchetto contiene anche un programmino ebb che estrae queste informazioni e le trascrive in file adeguati per l'uso che dvipdfm ne deve fare. Per i file PostScript in formato `.eps` e `.ps` esso ricorre automaticamente al processo di conversione attraverso l'invocazione del programma gs per ottenerne la versione `.pdf`. Questo ricorso al programma esterno è necessario perché dvipdfm è capace di interpretare codice PostScript sufficientemente semplice (per esempio quello contenuto nel file di uscita di METAPOST), ma non è in grado di interpretare codice PostScript arbitrariamente complesso. Questo è il motivo per il quale dvipdfm non è in grado di trattare file DVI che contengano le istruzioni PostScript generate dal pacchetto *PSTricks*.

Si segnala l'esistenza del programma dvipdfmx, un'estensione di dvipdfm, che dovrebbe presentare una maggiore funzionalità. Unito ad altri pacchetti di estensione, come per esempio *pdfcomment* consente di creare le *annotazioni*, una caratteristica del formato PDF, che non sarebbe nativamente realizzabile con il semplice latex. Tuttavia, salvo alcune limitazioni importanti, come l'impossibilità di convertire direttamente file che contengano codice PostScript avanzato, come quando si usa *PSTricks*, sembra che questo programma esteso sia il più efficace nel trasformare un file DVI nel corrispondente file PDF in termini di ingombro di memoria.

Metodo di compilazione	Ingombro in byte
pdftex	2 283 510
tex + dvipdfm	2 269 821
tex + dvipdfmx	2 007 012
tex + dvips + ps2pdf	3 485 081

Tabella 10.1: Dimensioni in byte dei file PDF ottenuti compilando il TEXbook con diversi metodi

A questo proposito merita di segnalare una tabellina, qui riprodotta in modo semplificato, generata da Péter Szabó e pubblicata negli atti della EuroTEX Conference del 2009, dove sono riportati gli ingombri in byte dei file PDF ottenuti con diversi procedimenti, tabella 10.1.

Infine si può mettere in evidenza un modo "pulito" per ottenere in uscita un file PDF direttamente da un file predisposto per essere compilato con latex e faccia uso dei comandi avanzati generati dal pacchetto *PSTricks*. Si tratta di usare il programma di compilazione ps4pdf; lo si può lanciare da linea di comando, ma si possono configurare diversi *shell editor* perché lo annoverino fra i motori di composizione che possono lanciare. Per TeXShop per esempio, basta aggiungere agli altri file `.engine` di cui già dispone nella cartella `~/Libreria/TeXShop/Engines/`, il file `ps4pdf.engine` contenente quanto segue:

```
#!/bin/tcsh

set path= ($path /usr/texbin /usr/local/bin)
ps4pdf  -file-line-error -shell-escape -synctex=1 "$1"
```

A questo file bisogna impostare il bit per poterlo eseguire dando da terminale il comando:

```
\chmod 755 ps4pdf.engine
```

Ciò fatto fra i motori di composizione che compaiono nel menù di TeXShop compare ora anche ps4pdf che può essere usato in alternativa agli altri programmi di composizione. Una impostazione del tutto simile si può fare per molti altri *shell editor*, naturalmente con le varianti specifiche di ciascuno di essi.

Si segnala che questo programma, pur essendo uno script che esegue in background diversi altri programmi del sistema TEX, non si limita a compilare il file DVI e a convertirlo in formato PDF con le procedure descritte sopra, ma estrae le parti scritte con i comandi e gli ambienti di *PSTricks*, le compila, le trasforma in formato PDF, le scontorna, e infine le rende disponibili ad una vera compilazione con pdflatex, il che implica che si possono avere nel file finale tutte le prestazioni di un file scritto direttamente con il linguaggio PDF, quindi, in particolare, si possono mantenere tutti i vantaggi della microgiustificazione. Il file sorgente del documento, infatti, può anche usare, per esempio, il pacchetto

pdfpages, di Andreas Matthias, che si ringrazia per la segnalazione; questo pacchetto può funzionare solo se il motore di composizione gestisce i comandi avanzati del linguaggio PDF.

10.3.2 I formati accettabili da pdflatex, lualatex e xelatex

pdflatex e gli altri programmi trattati in questo paragrafo, oltre al formato di uscita di METAPOST e al formato `.pdf`, per ovvi motivi di compatibilità, accettano in entrata anche i formati `.jpg` e `.png`. Con TeX Live a partire dal 2009 essi possono importare anche le figure in formato `.eps` convertendole automaticamente in formato `.pdf`. Dal 2010 TeX Live è attrezzato per eseguire direttamente la conversione senza bisogno di installare o invocare nessun ulteriore pacchetto di estensione. Di fatto, quindi, l'utente non ha bisogno di questi dettagli, ma può considerare questi programmi direttamente capaci di importare anche le figure EPS.

La figura 10.2 contiene infatti l'inclusione del file grafico `tiger.eps`, che fa parte dei file di esempio della distribuzione di GhostScript, convertito al volo durante l'esecuzione del programma pdflatex con cui è stato composto questo documento. Durante l'esecuzione si osserva una breve pausa, una tantum, durante la quale l'interprete pdftex si ferma un istante per lanciare un programma di sistema per convertire il formato `.eps` dell'immagine nel formato `.pdf`. Questa operazione è eseguita *una tantum*, nel senso che se il file convertito esiste già, l'interprete incorpora l'immagine convertita direttamente; se però l'immagine `.eps` è stata aggiornata con una versione più recente, la conversione viene eseguita di nuovo. I comandi per generare la figura 10.2[2] sono i seguenti:

```
\begin{figure}\centering
\includegraphics[width=.7\textwidth]{tiger.eps}
\caption{Inclusione del file \texttt{tiger.eps} da parte di
    \prog{pdflatex}}
\label{fig:tiger}
\end{figure}
```

Altrove e in versioni precedenti di questa Introduzione si è consigliato di non usare l'estensione dei file grafici; i motivi in parte erano e sono tuttora spiegati nella documentazione dei file *epsconversion* e *epstopdf*; ma dal 2010 ritengo che non si debba più seguire questo consiglio. Prima del 2010, infatti, nella stessa cartella potevano esistere sia un file `.eps` sia la sua conversione in un formato accettabile da pdflatex; il nome dei due file poteva essere lo stesso ma ne era diversa l'estensione. In questo modo, se non si specificava l'estensione, lo stesso file sorgente poteva essere elaborato sia da latex, che avrebbe letto e incluso solo i file `.eps`, sia da pdflatex, che avrebbe letto solo uno degli altri formati.

Ora, con la capacità di pdflatex di gestire i file `.eps` come se fossero nativi della sua natura, due file con lo stesso nome e diversa estensione possono non

[2]La figura `tiger.eps` viene distribuita insieme al programma `gs`.

Figura 10.2: Inclusione del file `tiger.eps` da parte di pdflatex

rappresentare più la stessa figura memorizzata in formati diversi, ma potrebbero rappresentare due figure distinte. Si commenta da sola l'inopportunità di distinguere le figure solo sulla base dell'estensione, ma non del nome; ma proprio per evitare confusioni e rendere più spedita la ricerca della figura da includere, ritengo che sia meglio specificare sempre anche l'estensione del file grafico che contiene la figura.

Si veda anche il paragrafo precedente in merito all'uso di ps4pdf, che consente di usare pdflatex anche quando il file da compilare usa il pacchetto *PSTricks* con i suoi comandi avanzati in linguaggio PostScript.

Dal 2012 è disponibile anche il pacchetto *svg* che permette di includere le immagini nel formato `.svg`; dal gennaio 2013 il pacchetto è disponibile nella distribuzione TEX Live 2012 ma, siccome si appoggia a programmi esterni, richiede l'attivazione indiscriminata dell'opzione *shell-escape* (o *enable-write18*) per poter agire su programmi esterni che non sono disponibili per tutte le piattaforme, per cui funziona solo sulle macchine Linux e Mac. Siccome il programma inkscape è disponibile per tutte le piattaforme, il pacchetto menzionato qui, che non è ancora operativo sulle macchine Windows, può venire ignorato su queste piattaforme purché si abbia la pazienza di fare a mano quello che il pacchetto *svg* è in grado di fare da solo. In sostanza sia usando il pacchetto, sia procedendo a mano si producono due file, uno in formato PDF che contiene l'immagine senza legende,

e un secondo file con il mark up LaTeX contenente le istruzioni per comporre e collocare al punto giusto le legende. Il tutto risulta molto comodo per eseguire disegni vettoriali con una interfaccia grafica interattiva, inkscape, che in certe circostanze può risultare di più comodo impiego rispetto all'uso dei programmi di disegno indicati in questa guida. È opportuno ricordare che l'output di inkscape è vettoriale, mentre quello di altri programmi di elaborazione grafica potrebbero non esserlo.

10.4 Conversione dei formati

Ci troviamo forse in un vicolo cieco? Per fortuna no. Basta poter convertire da un formato all'altro. Esistono diversi programmi per farlo e qui se ne elencano alcuni; la maggior parte sono disponibili per tutte le piattaforme. Inoltre, servendosi della distribuzione 2010 o successiva di TeX Live, molte delle conversioni elencate qui sotto diventano superflue quando si usi pdflatex, lualatex o xelatex; esse restano necessarie quando si debbano includere figure, per esempio in formato `.bmp` o in altri formati derivanti da macchine fotografiche particolari o da sistemi operativi diversi da quello col quale si sta lavorando. Qui di seguito si elencano solo programmi freeware gratuiti; ovviamente ne esistono anche di commerciali ottimi, si come il sistema TeX è freeware gratuito, così si cerca di usare la stessa logica per gli altri programmi.

ghostscript e, meglio ancora, la sua interfaccia grafica ghostview (conosciuta anche coi nomi gview o gv), permettono di trasformare una immagine `.ps` oppure `.eps` in immagine `.pdf` e viceversa. All'occorrenza il programma è capace di trasformare un file `.ps` in un file `.eps` assicurando che il file finale non contenga istruzioni PostScript incompatibili con la possibilità di inclusione in un altro file. ghostscript, tramite le opzioni che gli si possono specificare è anche in grado di trasformare i formati bitmapped nel formato EPS o PDF ma l'operazione è delicata e va fatta dopo aver consultato approfonditamente la documentazione di ghostscript.

epstopdf epspdf e ps2pdf, o altri simili applicativi da linea di comando, eseguono le analoghe trasformazioni senza interfaccia grafica; spesso hanno il vantaggio di corredare il file di uscita dell'informazione corretta in merito al 'bounding box', cioè alle dimensioni del rettangolo circoscritto all'immagine effettiva. Con la distribuzione 2010 di TeX Live non è necessario ricorrere a programmi esterni, nel senso che ci pensano direttamente programmi di composizione a eseguire la conversione senza che l'utente debba preoccuparsene. Bisogna ricordare che la conversione dal formato `.eps` al formato `.pdf` talvolta *non* comporta l'inclusione dei font eventualmente usati nel file di partenza; questo fatto può portare a certi inconvenienti, quando il file `.pdf` viene incorporato in un file principale che verrà letto su un'altra macchina che non dispone degli stessi font; oppure rende il file principale incompatibile con il formato archiviabile (vedi il capitolo 22).

eps2pdf è una interfaccia grafica disponibile solo per macchine Windows che esegue sostanzialmente lo stesso tipo di trasformazione di epstopdf, ma con il vantaggio dell'interfaccia grafica.

eps2png e eps2jpg sono programmi da linea di comando per macchine Linux/UNIX; trasformano le immagini vettoriali in formato `.eps` in immagini a matrici di punti diversamente compresse nei formati `.png` oppure `.jpg`. In generale sarebbe desiderabile evitare questi tipi di conversione, ma è opportuno limitarsi alle conversioni eseguibili con programmi descritti nei punti precedenti che conservano la qualità vettoriale dell'immagine. Tuttavia, in vista di produrre file archiviabili, talvolta questa potrebbe essere una soluzione per trasformare file `.pdf` incompatibili con l'archiviabilità in file compatibili, in quanto i "disegni" dei font vengono trasformati in "disegni" a matrici di pixel; certo se sulla macchina dove si esegue la trasformazione mancano i font scalabili, anche questo tipo di trasformazione diventa inutile.

pdftops esegue da linea di comando la conversione dal formato `.pdf` al formato `.ps`. Anche se il formato `.pdf` è sempre da preferire, talvolta può rendersi necessario ricorrere a questo tipo di trasformazione.

jpegtops trasforma un file `.jpg` in formato `.eps`. Si tratta di un applicativo da linea di comando. Assicura che il bounding box sia specificato correttamente. In realtà questo programma si limita ad avviluppare il codice compresso dell'immagine a matrice di pixel dentro un involucro PostScript, che contiene l'informazione corretta del 'bounding box', ma per il resto si limita a definire dove inizia e dove finisce il codice dell'immagine compressa.

mptopdf esegue la trasformazione di un file prodotto con METAPOST in un file `.pdf`. Questa trasformazione in effetti non sarebbe necessaria se si dispone di un sistema TEX moderno e completo; tuttavia se si dovesse mandare la figura a qualcun altro di cui non si conoscono le dotazioni informatiche, è meglio mandargliela in formato PDF piuttosto che nel formato di uscita di METAPOST, perché il destinatario potrebbe non avere familiarità con questo formato, oppure potrebbe mancare del software necessario per visualizzarla o per includerla nei suoi documenti.

Vale la pena di segnalare che lo *shell editor* dei sistemi MacOS X, TeXShop, consente anche di eseguire la compilazione di un file METAPOST, ma non si limita ad ottenere il file in codice PostScript, bensì provvede ad eseguire direttamente il programma mptopdf cosicché il risultato finale è direttamente il file PDF contenente il disegno perfettamente ritagliato al rettangolo tangente. Anche alcuni altri *shell editor*, certamente lo shareware WinEdt per macchine Windows, consentono di eseguire la compilazione dei disegni METAPOST; forse qualcuno consente anche la trasformazione automatica nel formato PDF.

Paint e i programmi simili aprono i file bitmapped e li trasformano in qualunque altro file a matrici di pixel, sia `.png` sia `.jpg`; quindi per gli altri formati bitmapped non dovrebbero esserci problemi per ottenere formati compatibili con i programmi di nostro interesse.

gimp si trova per tutte le piattaforme; è un formidabile programma di editing grafico che permette di eseguire tutte le trasformazioni dell'immagine e la loro conversione in un qualunque formato a matrici di pixel; riesce a farlo anche con input vettoriali, ma salva solo in formati a matrici di pixel. È un programma fortemente consigliabile per chiunque e su qualunque piattaforma. Sarebbe però consigliabile non usarlo per convertire formati vettoriali in formati a matrici di pixel, limitandosi alle altre trasformazioni e all'editing degli altri formati.

ImageMagik è un altro noto programma di editing grafico; esso mette a disposizione dell'utente il comando convert che permette di eseguire rapidamente la conversione da un formato grafico all'altro; viene consigliato e, per esempio, viene suggerito di definire la regola grafica

```
\DeclareGraphicsRule{.tif}{png}{.png}%
{‘convert #1 ‘basename #1 .tif‘.png}
```

per convertire le immagini `.tif` in immagini `.png`. Questo metodo, in realtà, può venire usato su qualunque piattaforma e richiede l'abilitazione all'esecuzione di programmi esterni mediante l'opzione della linea di comando `-shell-escape`, della cui 'pericolosità' si è già parlato.

10.5 Scontornare le immagini

Un aspetto che si dimentica troppo spesso è quello di scontornare le immagini. LaTeX lascia da solo gli spazi bianchi necessari attorno alle immagini; se queste a loro volta contengono altri spazi bianchi nel loro contorno, nel documento composto risulterà esserci alla fine troppo spazio bianco. Non è semplice poter scontornare le immagini, specialmente se ci si vuole attenere a programmi freeware.

gimp consente di scontornare e di correggere qualunque formato grafico ma permette di salvare solo nei formati a matrici di pixel; non è quindi il caso di servirsi di gimp per scontornare immagini vettoriali.

gview permette di correggere il bounding box, cosicché di fatto si scontorna l'immagine. In ogni caso questo programma permette di visualizzare sullo schermo il rettangolo tangente, cosicché diventa facile spostare il cursore a croce sul vertice inferiore sinistro e successivamente sul vertice superiore destro per rilevare nell'apposito spazio della cornice inferiore del

programma le coordinate di questi due vertici; queste coordinate potranno essere usate come quelle del 'bounding box' quando si usa il comando `\includegraphics` per includere l'immagine.

Preview è un applicativo del sistema operativo MacOS X[3]; serve a molte funzioni, in particolare per visualizzare i file PDF, JPEG, PNG, TIFF e GIF; fra le operazioni che riesce a fare su questi file c'è anche l'operazione di *cropping*, cioè di scontornare le immagini mediante una comoda interfaccia grafica; permette anche di salvare ogni file in uno degli altri formati che sa gestire. Ovviamente è usabile solo con quel sistema operativo.

pdfcrop è un programma (uno script in Perl) che permette di scontornare molto bene le immagini in formato PDF; esso è distribuito con il sistema TEX, quindi è già disponibile sulla piattaforma di lavoro e non è necessario ricorrere ad installazioni particolari. Questo programma usa ghostscript per determinare il rettangolo circoscritto e per ritagliare via i bordi bianchi salvando il file smarginato con un altro nome; può essere usato dall'utente usando la finestra comandi, Meglio ancora, si può smarginare una figura al volo attraverso pdfcrop usando il pacchetto *standalone*, alla cui documentazione si rimanda

Ma non tutto è perduto; se non si riesce a scontornare con programmi adeguati, si può sempre operare dall'interno dei programmi di composizione del sistema TEX usando correttamente il comando `\includegraphics` con le sue opzioni accompagnate dai loro valori.

10.6 L'importazione delle immagini

Se il lettore non si è scoraggiato con queste (apparenti) difficoltà e ha già (all'occorrenza) convertito i formati delle immagini in uno dei formati consentiti, e le ha già scontornate, allora può procedere con l'effettiva importazione delle immagini.

Basta richiamare il pacchetto *graphicx* con il solito comando

```
\usepackage{graphicx}
```

e usarne i comandi e le opzioni in modo corretto.

Nel momento i cui si deve importare una figura esterna basta usare con la seguente sintassi il comando `\includegraphics` (generalmente, ma non necessariamente, all'interno dell'ambiente *figure*):

`\includegraphics[`⟨*lista delle chiavi*⟩`]{`⟨*file*⟩`}`

[3]Nella versione italiana di MacOS X, questo programma si chiama Anteprima.

Il ⟨*file*⟩ è il nome del file contenente l'immagine *preferibilmente con l'estensione specificata*; se la figura con il formato giusto esiste, il compilatore importa la figura; altrimenti il programma emette un messaggio di errore ma consente di proseguire, evidentemente senza importare nulla. Ovviamente se l'estensione è specificata in modo errato, il programma non trova la figura nel formato giusto e non importa nulla. Se però due diverse figure risiedessero nella stessa cartella con lo stesso nome ed estensione diversa, il programma potrebbe trovare e importare ciò che ha trovato, ma potrebbe essere una figura che non c'entra niente. Non si metterà abbastanza in evidenza l'importanza di gestire i propri file, testuali o grafici che siano, in modo intelligente al fine di evitare simili situazioni spiacevoli.

Se invece è stata definita una regola grafica, per esempio quella mostrata per convertire le immagini dal formato `.tif` al formato `.png`, allora la prima volta che si esegue la compilazione viene ricercata anche l'estensione da convertire, viene eseguita la conversione e viene usato il file convertito; nelle eventuali compilazioni successive il programma trova direttamente il file nel formato giusto e non ha più bisogno di eseguire nessuna conversione. Questo vale anche per le figure in formato `.eps` trasformate da `pdflatex` stesso in file con l'estensione `.pdf`.

Per i file in formato `.svg` il pacchetto *svg* mette a disposizione il comando `\includesvg` con la stessa sintassi del comando `\includegraphics`; naturalmente la ⟨*lista delle chiavi*⟩ ammette alcune chiavi specifiche per il formato `.svg` che non possono essere usate con il comando `\includegraphics`; il lettore è rinviato alla documentazione del pacchetto *svg* leggibile, come al solito, con il comando da terminale `texdoc svg`.

10.6.1 Organizzare le immagini

Vale la pena di esporre un dettaglio organizzativo, che però presenta vantaggi e svantaggi e va usato con intelligenza; per la gestione delle figure non ci sono problemi se i file che le contengono sono nella stessa cartella del file sorgente da comporre; tuttavia questo modo di procedere è disordinato e alla fine la cartella contiene tante cose disparate e non si riesce più a gestire. La soluzione è quella di dedicare/creare una sotto cartella dove custodire le immagini e solo loro. Supponendo di scrivere un libro su Paperino, Topolino e Pluto, probabilmente il file sorgente si troverà nella cartella `PTP/`; in questa cartella si trova la sottocartella `foto/` (se non c'è la si crea); in questa sottocartella si trova la fotografia `Topolino.jpg`. Allora questa foto può essere inclusa per esempio specificando il percorso relativo per raggiungerla:

```
\includegraphics{foto/Topolino}
```

Più comodo e meno suscettibile di errori è invece il procedimento di specificare il percorso (*relativo alla cartella dove si trova il file sorgente*) dell'unica o delle varie cartelle, dove si trovano le immagini, mediante il comando `\graphicspath` con la seguente sintassi:

```
\graphicspath{{⟨path1⟩}{⟨path2⟩}...{⟨pathn⟩}}
```

dove è opportuno notare le parentesi graffe esterne che racchiudono le coppie di graffe interne, ognuna delle quali specifica un diverso percorso. I comandi

```
\graphicspath{{foto/}}
...
\includegraphics{Topolino}
```

permettono di includere la foto di Topolino senza bisogno di specificarne il percorso. Se le immagini relative a Paperino si trovassero in un'altra sottocartella, per esempio `fotoPaP/`, allora basterebbe specificare

```
\graphicspath{{foto/}{fotoPaP/}}
```

per avere accessibili entrambe le sottocartelle senza bisogno di specificarle per ogni inclusione[4].

Quali sono gli inconvenienti di questo metodo? Le ragioni sono ben nascoste nelle macro che ricercano i file e nel modo di accedervi. Innanzi tutto vale la pena di sottolineare che si possono dare in successione, anche in file diversi molti comandi `\graphicspath` con una lista di percorsi formati da un solo percorso o da un certo numero di percorsi.

I successivi comandi definiscono i loro percorsi aggiornando una lista conservata dentro le viscere dei programmi in uso; e questa lista può essere anche piuttosto lunga, quando con un unico comando si specificano tutti i percorsi di ricerca per l'intero documento.

Quando la lista contiene tutti i percorsi di ricerca e `\includegraphics` deve includere un file grafico come, per esempio, `Topolino.jpg`, antepone ricorsivamente al nome di questo file i nomi dei percorsi memorizzati; appena trova il file, interrompe la ricorsione, e agisce sul file. Se non si arresta prima perché il file è stato trovato, la ricorsione è condotta fino all'ultimo elemento della lista; ognuna di queste ricerche consuma tempo sia per essere eseguita sia per generare in modo automatico la concatenazione di ciascuno dei percorsi con il nome del file; se questo non avesse l'estensione, la ricerca verrebbe ripetuta anche concatenandovi ricorsivamente tutte le estensioni valide per il programma specifico di composizione.

[4] In verità prima del 2010 sarebbe stato possibile specificare anche dei percorsi assoluti, invece che relativi alla cartella del file principale. Tuttavia anche prima del 2010 qui si consigliava di evitare questa possibilità, specialmente quando il documento deve essere composto a più mani, su macchine diverse, con sistemi operativi diversi; in queste condizioni è preferibile evitare qualunque specificazione che sia dipendente dalla particolare macchina e dal particolare sistema operativo. La portabilità dei file `.tex` è una delle caratteristiche del sistema TEX e sarebbe quanto mai inopportuno inficiare questa particolarità. Dal 2010 non si possono più specificare percorsi assoluti, quindi il problema non si pone più. Conviene ricordare ancora che anche sulle macchine Windows dove il separatore delle cartelle è il backslash, quando si specificano i percorsi all'interno di programmi con il mark-up TEX, LATEX, eccetera, il separatore delle cartelle deve essere la barra normale e ci pensa il programma di composizione a convertirlo nel backslash quando deve aprire o salvare dei file.

Se si usasse quindi `\graphicspath` con 25 path diversi per includere le foto di 25 capitoli, ognuno con le sue foto in una cartella diversa e le estensioni possibili fossero quattro, un file con la quarta estensione, da includere nell'ultimo capitolo, il 25°, verrebbe ricercato 100 volte; la foto verrebbe trovata al centesimo tentativo, sempre che il nome fosse corretto, la foto ci fosse per davvero e con una delle estensioni accettabili, eccetera. Se invece si fosse specificato semplicemente il nome del file completo di estensione e preceduto dal suo percorso relativo in modo esplicito, il file sarebbe stato trovato con una sola ricerca, e non sarebbe stato necessario ripeterla; al massimo se il percorso o il nome o l'estensione fossero stati specificati in modo errato, il programma avrebbe emesso un messaggio d'errore e l'operatore avrebbe potuto sapere subito in che cosa consisteva l'errore e avrebbe potuto poi correggerlo adeguatamente.

Non si abbia quindi paura a conservare le immagini di ogni capitolo in cartelle diverse, subordinata alla cartella che contiene il file sorgente del capitolo; la specificazione dell'intero percorso per ciascuna figura può essere noiosa, ma rende il file sorgente molto più facile da leggere e da correggere. Alternativamente si dia un nuovo comando `\graphicspath` all'inizio di ciascun capitolo per specificare l'unico percorso per trovare le immagini relative a quel capitolo.

Nello stesso modo la soluzione dell'uso di `\graphicspath` è molto conveniente quando le figure da includere non sono in numero sterminato e possono essere conservate in modo ordinato in una sola cartella.

Un'ultima precisazione; nei sistemi UNIX (Linux, Mac) i link simbolici sono molto usati; nei sistemi Windows essi sono disponibili solo a partire dal sistema operativo Windows Vista; quindi sono una novità abbastanza recente e non ancora entrati nell'uso "quotidiano" degli utenti Windows.

Un link simbolico è una indicazione che appare in una specifica cartella del sistema operativo, ma che "punta" ad un'altra cartella o ad un file presente in un'altra cartella.

Nei sistemi UNIX si possono creare link simbolici usando il terminale, ponendosi nella cartella dove si vuole creare il link e dando il comando

`ln -s` ⟨*file o cartella puntata completi di percorso*⟩ ⟨*nome del link*⟩

Nei sistemi Windows li si crea in modo analogo, usando la finestra comandi (o prompt dei comandi), mettendosi nella cartella dove si vuole creare il link e dando il comando:

`mklink` ⟨*nome del link*⟩ ⟨*file o cartella puntata completi di percorso*⟩

È opportuno, ma non necessario, che il nome del link abbia le stesse caratteristiche apparenti del file o della cartella puntata; per esempio il solo nome basterà per una cartella, ma per puntare al nome di una immagine in formato `.jpg`, è meglio che il nome del link abbia l'estensione `.jpg`. Se si cancella il link con i soliti comandi, si cancella solo il link, non l'oggetto puntato; per il resto qualunque altra operazione eseguita sul link è come se la si facesse sull'oggetto puntato.

Usando i link si può aggirare la proibizione di usare indirizzi di file che non si trovino nella cartella del file principale o in una delle sue sottocartelle. Si pensi alla immagini contenute in album già organizzati e residenti altrove sul disco fisso; un link simbolico posto in una cartella subalterna a quella del file principale, può rendere accessibile l'intero album come se lo si fosse copiato nella cartella subalterna.

Però usando i link simbolici l'insieme di cartelle che costituiscono il documento non sono più compatibili con altre macchine, indipendentemente dal sistema operativo, se non hanno esattamente lo stesso contenuto in ogni ramificazione degli alberi di cartelle a partire dalla radice del disco rigido (almeno quelle ramificazioni che coinvolgono il documento). I link simbolici possono essere utili solo per un lavoro "solitario", non in collaborazione con altri coautori, a meno che l'assemblaggio dei vari contributi non sia eseguito da un solo curatore.

10.6.2 Includere le immagini

Per includere le immagini, come si è già detto sopra, bisogna usare il comando `\includegraphics` e bisogna specificargli correttamente la ⟨*lista delle chiavi*⟩.

Questa è costituita da una serie di dichiarazioni separate da virgole e scritte nella forma 'chiave = valore'. Se la chiave rappresenta una affermazione booleana, non è necessario specificarne il valore 'true' ma, quando la si vuole usare con il valore 'false', è necessario specificarlo.

Qui non si esporranno tutte le opzioni disponibili; ci si limiterà a descrivere quelle più comunemente usate, almeno secondo l'esperienza dello scrivente; si rimanda il lettore alla documentazione del pacchetto (`texdoc grfguide`) per avere maggiori dettagli e per le informazioni relative alle altre opzioni.

`bb` Serve per specificare il rettangolo tangente all'immagine con la sintassi:

`bb=` ⟨*llx*⟩ ⟨*lly*⟩ ⟨*urx*⟩ ⟨*ury*⟩

In generale non è necessario specificare le coordinate dei vertici inferiore sinistro (⟨*llx*⟩: *lower left x* → (ascissa [vertice] inferiore sinistro); ⟨*lly*⟩: *lower left y* →(ordinata [vertice] inferiore sinistro)) e superiore destro (⟨*urx*⟩: *upper right x* → (ascissa [vertice] superiore destro); ⟨*ury*⟩: *upper right y* →(ordinata [vertice] superiore destro)) del rettangolo tangente; queste coordinate in generale sono contenute nel file grafico da importare. Tuttavia queste informazioni potrebbero mancare oppure potrebbero non corrispondere a quelle del rettangolo tangente. Un caso molto frequente si ha quando con un programma grafico esterno al sistema TEX si crea o si ottiene un disegno o una immagine a mezze tinte da scontornare perché magari il disegno appare su una pagina in formato A4, non su un rettangolino grande quanto il disegno; oppure l'immagine fa parte di una pagina che contiene anche altre parti grafiche che non interessa importare. Si può usare il pacchetto *`standalone`* per smarginare le figure,

come si è spiegato in precedenza. In alternativa si può usare la chiave `clip` (o, alternativamente, il comando asteriscato `\includegraphics*`) insieme a una delle chiavi `viewport` o `trim` che verranno descritte poco oltre.

`width` serve per specificare la larghezza dell'immagine nel documento finale; generalmente conviene parametrizzare questa dimensione ad un valore legato alla geometria della gabbia di stampa, invece di specificare un numero di punti, o di millimetri, o di pollici, o di[5]...

Allora è meglio specificare la larghezza nella forma `width=0.5\linewidth` invece di `width=87mm`, perché cambiando layout della pagina la dimensione relativa mantiene le proporzioni, mentre la dimensione assoluta potrebbe dare luogo a inconvenienti non lievi; si pensi per esempio di voler passare da una composizione a piena pagina ad una composizione a due colonne; 87 mm potrebbero andare bene a piena pagina, ma potrebbero essere troppi per una colonna.

Vale la pena di ricordare che `\linewidth` coincide con la giustezza corrente (piena pagina o colonna) salvo che all'interno di certi ambienti potrebbe essere un poco inferiore; è quindi conveniente fare sempre riferimento a `\linewidth` invece che a `\textwidth` o a `\columnwidth`.

Secondo una tradizione conservata dalle norme UNI, i fattori da preferire per moltiplicare la larghezza della linea sono 1, 0,5, 0,2 e le loro radici quadrate. Per le immagini i fattori più piccoli sono da evitare, ovviamente, ma la sequenza di ragione (approssimativa) $\sqrt{2}$ è quella più frequente.[6]

`height` serve per specificare l'altezza dell'immagine riprodotta nel documento; anche in questo caso è meglio fare riferimento all'altezza della gabbia di stampa, parametrizzando rispetto a `\textheight`; per esempio, conservando uno dei fattori di scala della sequenza UNI, si potrebbe specificare: `height=0.35\textheight`.

`keepaspectratio` Se si sono specificate sia `width` sia `height` la figura potrebbe subire cambiamenti di scala diversi in altezza rispetto alla larghezza; per evitare questo fatto la variabile booleana `keepaspectratio` mantiene inalterato il rapporto di forma dell'immagine da riprodurre e sceglie sia per l'altezza sia per la larghezza il maggiore fra i due rapporti di scala che *non* fa eccedere nessuna delle due dimensioni specificate.

`viewport` serve per specificare la finestra attraverso cui guardare l'immagine; la finestra in effetti fa sì che diventi un nuovo rettangolo tangente all'immagine

[5] Le unità di misura dimensionali che il sistema TeX capisce sono i punti (pt), i millimetri (mm), i centimetri (cm), i pollici (in) oltre alle unità legate alle dimensioni del font corrente; l'altezza della 'x' (ex), la larghezza di una 'M' (em). TeX capisce anche altre unità meno frequenti e dall'uso un po' particolare, utili specialmente per chi scrive pacchetti di estensione o file di classe.

[6] In sostanza i fattori di scala da preferire per le immagini sono quelli (arrotondati) della sequenza geometrica di ragione $\sqrt{2}$ che contiene il valore unitario: 0,35, 0,5, 0,7, 1, 1,4, 2, 2,8.

e, insieme a `clip`, può servire per nascondere le parti dell'immagine che non interessano; quindi può servire per scontornare una immagine alla quale non è stato possibile applicare il trattamento specificato nei paragrafi precedenti. Tuttavia può servire anche per mostrare solo un particolare dell'immagine. La sua sintassi è la seguente:

`viewport=` ⟨*llx*⟩ ⟨*lly*⟩ ⟨*urx*⟩ ⟨*ury*⟩

dove ⟨*llx*⟩ ⟨*lly*⟩ sono le coordinate x e y dell'angolo in basso a sinistra della finestra e ⟨*urx*⟩ ⟨*ury*⟩ sono le coordinate x e y dell'angolo in alto a destra della finestra. Queste coordinate sono relative al rettangolo circoscritto originario all'immagine, e questo potrebbe non coincidere con il supporto virtuale sul quale l'immagine digitale è riportata. Non è il caso di preoccuparsi di questo dettaglio perché il 99% delle volte si ha assoluta coincidenza.

`trim` serve ad una funzione simile a quella prodotta da `viewport`, solo che i quattro valori sono le ampiezze delle strisce di immagine da togliere successivamente da sinistra, dal basso, da destra e dall'alto dell'immagine; talvolta è più facile specificare questi valori, che non le coordinate della finestra attraverso cui vedere una parte dell'immagine. La sintassi è la seguente:

`trim=` ⟨*left*⟩ ⟨*bottom*⟩ ⟨*right*⟩ ⟨*top*⟩

Sia con la chiave `viewport` sia con `trim` le dimensioni sono espresse indicando le unità di misura che TEX gestisce; il programma provvede a trasformarle in *punti PostScript*, visto che l'operazione viene svolta in quel linguaggio vero o semplificato; se si desidera esprimere quelle lunghezze direttamente in punti PostScript si usa l'unità di misura `bp`, oppure non si specifica l'unità di misura che per default viene quindi assunta uguale a `bp`.

`clip` serve per dare l'ordine di tagliare effettivamente quanto sporge fuori del rettangolo tangente specificato dalla chiave `bb`, oppure da `trim`, oppure da `viewport`. Se non si esprimesse `clip`, il programma tratterebbe l'immagine come se le sue dimensioni effettive fossero quelle del rettangolo specificato con le chiavi suddette, ma quanto sporgesse fuori da quel rettangolo non verrebbe tagliato via e quindi l'immagine sarebbe ancora tutta integra ma più grande di quanto uno si aspetterebbe. Se invece di `\includegraphics` si usasse la versione asteriscata `\includegraphics*`, allora non sarebbe necessario specificare la parola chiave `clip`, perché il valore `true` con il comando asteriscato è quello di default.

`angle` serve per specificare l'angolo di rotazione (in gradi e in senso antiorario) della figura; quando si usa questa specifica (indicando generalmente $\pm 90°$

o 180°) insieme a indicazioni di scala del tipo `width=...` bisogna stare attenti a quale operazione si indica per prima e perciò si esegue per prima; normalmente è conveniente specificare per prima la rotazione, perché è proprio quello che si vuole ottenere.

Bene inteso è possibile specificare anche altre chiavi e i loro valori, ma a quel punto è meglio andare a leggere direttamente la documentazione `grfguide.pdf` che accompagna sempre il sistema TeX e si legge con `texdoc grfguide`.

Nella figura 10.3 sono rappresentati alcuni effetti che si possono ottenere con la stessa fotografia; sotto ogni foto è indicata la lista delle chiavi usata per ottenere il risultato. Nella prima foto in alto a sinistra si ha l'immagine a larghezza piena, fatta alla manifestazione Cheese, patrocinata da Slow Food, che si svolge ogni due anni a Bra (CN); nelle immagini successive si sono applicate diverse chiavi e, in particolare, quando sono state specificate sia `width` sia `height` senza specificare `keepaspectratio`, si è ottenuta una deformazione dell'immagine in senso orizzontale che è decisamente visibile ad occhio nudo. Nelle due immagini dove si è specificato o `trim` o `viewport` insieme alla dichiarazione `clip` l'immagine risulta correttamente ritagliata e ingrandita fino a soddisfare il requisito dell'altezza. Nelle ultime immagini la foto del Minareto di Marrakech "La Koutoubia" è stata ripresa con la fotocamera ruotata di 90° (be', ecco, ehm, circa...) per cui la corretta rappresentazione richiede una rotazione contraria; appare evidente che nella foto di sinistra la base è stata scalata fino ad essere uguale alla larghezza della riga, poi l'immagine è stata ruotata così che la base diventi l'altezza; invece nella foto di destra prima l'immagine è stata ruotata poi la base è stata scalata alla larghezza della riga e solo in questo modo si è ottenuto il risultato desiderato.

È bene notare che le operazioni eseguite con le chiavi `trim` e `viewport` non sembrano così semplici da fare perché non sono generalmente note le dimensioni naturali dell'immagine; tuttavia è bene ricordare che le dimensioni del rettangolo tangente oppure del rettangolo che deve funzionare da finestra sono sempre rilevabili con il cursore a croce del programma gview, se l'utente ne dispone.

In mancanza di tale programma, vale la pena di ricordare che ogni volta che si lancia uno dei programmi di composizione del sistema TeX, viene prodotto un file con estensione `.log` e con il nome uguale a quello del file sorgente che si è appena compilato. In questo file (un normale file testuale contenente solo caratteri ASCII, quindi leggibile senza problemi con qualunque programma di elaborazione testi) quando viene importato per la prima volta un dato file grafico contenente un'immagine, ne viene scritto il tipo e ne vengono fornite le dimensioni 'base per altezza' della 'bounding box' in punti tipografici; con queste dimensioni è possibile fare le debite proporzioni di ciò che si desidera ritagliare via e quindi non è difficile trovare i valori numerici da usare; ricordiamoci infatti che le coordinate da specificare sono quelle del file di partenza, non del file riprodotto, quindi è necessario riferirsi alle coordinate del file di partenza.

`width=\linewidth`

`height=0.5\linewidth`

`width=\linewidth,`
`height=0.5\linewidth`

`width=\linewidth,`
`height=0.5\linewidth,`
`keepaspectratio`

`height=0.5\linewidth,`
`trim=100 50 150 120, clip`

`height=0.5\linewidth,`
`viewport=100 50 492 362, clip`

`width=\linewidth, angle=90`

`angle=90, width=\linewidth`

Figura 10.3: Due foto trattate con diverse chiavi

Capitolo 11

La bibliografia

Questo capitolo è stato rivisto completamente da Filippo Vomiero, autore anche della guida tematica *Comporre la bibliografia in LaTeX— `biblatex` e i software di gestione bibliografica* sia nella sua veste di utente di queste componenti del sistema TeX, sia nella sua veste professionale di bibliotecario; chi meglio di lui potrebbe conoscere questi argomenti visto che vi lavora quotidianamente? Il lettore che voglia approfondire l'argomento, lo può fare attraverso la lettura della sua guida tematica nel sito del GuIt.

11.1 Introduzione

Come si è già detto, LaTeX compone la bibliografia come un testo speciale costituito da un elenco di voci etichettate con un'etichetta di vari stili, ma citabili con una chiave mnemonica scelta dall'autore o fissata in altro modo.

Al paragrafo 7.5 è stato mostrato come preparare un elenco con la giusta sintassi per l'ambiente *thebibliography*, ma è altrettanto importante elencare i riferimenti in un ordine logico, scrivere tutte le informazioni necessarie al reperimento dell'opera citata, e, specialmente, essere coerenti nello stile di presentazione di tutte queste informazioni.

Vale la pena di ricordare che le bibliografie e gli elenchi bibliografici sono regolati dalla norma ISO 690:2010, [32], adottata in Italia con la norma UNI ISO 690:2014. Queste norme specificano quali siano le informazioni necessarie e quelle facoltative richieste per un riferimento correttamente conforme alle norme. In buona sostanza sono tutte le informazioni che permettono di identificare univocamente il riferimento citato. Le norme non specificano in che stile esse debbano essere stampate; l'importante è che informazioni dello stesso tipo siano stampate con lo stesso stile. In sintesi, quando si scrive una bibliografia, le due parole chiave da tenere a mente sono: *identificazione*, ovvero gli elementi forniti devono essere sufficienti per distinguere un documento da un altro, e poterlo recuperare dal catalogo di una biblioteca o di una banca dati, eventualmente

individuando il segmento specifico a cui si fa riferimento nel testo; *coerenza*, vale a dire che una volta operata una scelta sulle regole da seguire (lo stile), esse vanno applicate rigorosamente in tutta la bibliografia.

Quando si utilizza l'ambiente *thebibliography*, per ciascun riferimento bibliografico, è necessario seguire scrupolosamente —e manualmente!— le regole dello stile adottato per quanto concerne l'ordine dei vari elementi, la forma e la punteggiatura. Ogni modifica successiva (ad esempio, nell'ordine in cui compaiono le citazioni, oppure se ci viene richiesto di usare un altro stile) richiede lunghe e tediose riscritture: se la bibliografia consiste anche soltanto di poche decine di voci, le possibilità di commettere errori, ma soprattutto, di non essere coerenti sono numerosissime, e il risultato finale non sarà all'altezza del tempo impiegato.

È molto più pratico inserire le informazioni bibliografiche in un database esterno; LaTeX tramite alcuni programmi specifici (BibTeX e biber) può recuperare agevolmente tali informazioni, e con l'uso di pacchetti appositi, come *natbib* o *biblatex* è possibile costruire automaticamente le citazioni bibliografiche e le bibliografie potendo scegliere tra un'ampia varietà di stili.

Il consiglio che vogliamo dare è di utilizzare i database bibliografici, anche se l'operazione all'inizio può sembrare complicata. È necessario innanzitutto un cambio di approccio al riferimento bibliografico: nell'ambiente *thebibliography* le informazioni vengono inserite in maniera *discorsiva*, ovvero in una o più righe vengono introdotte di seguito tutte le informazioni pertinenti. Questa è la forma a cui siamo più abituati, poiché tutte le bibliografie che abbiamo incontrato sono scritte così. Un database invece chiede di inserire le informazioni in maniera *classificata*, ovvero ciascuna informazione è associata a un identificatore univoco che ne esplicita la categoria (titolo, autore, ecc.).

Naturalmente per le varie piattaforme esistono delle interfacce grafiche che consentono di comporre i database bibliografici nel modo corretto; tanto per citarne uno, per tutte le piattaforme esiste il programma in linguaggio Java JabRef, che è un programma molto completo, supporta database sia nel formato classico di BibTeX che in quello più recente utilizzato da *biblatex*, e, cosa molto utile, è in grado di segnalare quali sono i campi obbligatori e quali i facoltativi, a seconda del tipo di materiale, nel rispetto della norma ISO 690. Per Mac OS X esiste (e accompagna il sistema TeX con l'installatore specifico MacTeX il programma BibDesk, che fa più o meno le stesse cose. Inoltre diversi shell editor, come per esempio TeXstudio, consentono di creare direttamente il database bibliografico.

11.2 I database bibliografici

Un database bibliografico è un file di testo con estensione `.bib`; a seconda del programma di estrazione bibliografica e dei pacchetti che si decide di utilizzare, è importante decidere anche l'input encoding del database.

Quando si utilizza BibTeX, per la trasportabilità sarebbe meglio che si trattasse di caratteri ASCII puri (codici a 7 bit) e quindi i segni letterali con

diacritici dovrebbero essere rappresentati con le sequenze per gli accenti che non sono state descritte finora da nessuna parte, ma si rinvia a qualunque guida o manuale 'oldstyle' di LaTeX[1]; negli anni più recenti sono state create due versioni di BibTeX con supporto esteso alle codifiche dei caratteri: bibtex8 e bibtexu, che permettono di utilizzare, rispettivamente, codifiche a 8 bit e Unicode. Allo stato attuale, tuttavia, è preferibile usare biber, che oltre al supporto nativo di Unicode, ha una migliore gestione delle lingue, e la creazione di stili personalizzati, grazie al pacchetto *biblatex* da usare assieme, è molto facilitata.

Dopo questa lunga premessa, veniamo al database; ogni voce del database comincia con la dichiarazione del tipo di documento a cui la voce si riferisce. In BibTeX questa dichiarazione è una delle seguenti parole: `article`, `book`, `booklet`, `conference`, `inbook`, `incollection`, `inproceedings`, `manual`, `masterthesis`, `misc`, `phdthesis`, `proceedings`, `techreport`, `unpublished`. Ognuna di queste categorie di documenti è spiegata nella documentazione di BibTeX, consultabile con `texdoc btxdoc`.

Se si utilizza biber assieme al pacchetto *biblatex*, sono disponibili ulteriori definizioni: `mvbook`, `bookinbook`, `suppbook`, `collection`, `mvcollection`, `suppcollection`, `dataset`, `online`, `patent`, `periodical`, `suppperiodical`, `mvproceedings`, `reference`, `mvreference`, `inreference`, `software`. Anche in questo caso si rimanda alla documentazione di *biblatex* per una spiegazione più approfondita (consultabile con il comando `texdoc biblatex`).

In base alle norme ISO ognuna di queste tipologie di documento richiede certe informazioni, quindi nel descrivere il documento bisogna usare un certo numero di parole chiave; se per un certo tipo di riferimento una voce obbligatoria manca, quando viene eseguito il programma di estrazione bibliografica, vengono emessi degli avvertimenti del fatto che la citazione tale è priva della specificazione talaltra. Le chiavi facoltative e obbligatorie per ogni tipologia sono specificate nella rispettiva documentazione citata.

Qui ci limiteremo a qualche piccolo esempio commentando ciò che appare nelle voci del database.

```
@manual{man:refbib,
  title=       {Information and documentation -- Guidelines for
                bibliographic references and citations to
                information resources},
  note=        {Norma {ISO} 690--2010},
  organization={International Organization for Standardization},
  address=     {Ginevra},
  year=        {2010},
  key=         {ISO 690}
}

@manual{man:patashnik,
```

[1] Si veda comunque il paragrafo 29.2.5 e la tabella 29.1.

```
  author=       {Oren Patashnik},
  title=        {\textsc{Bib}{\TeX}ing},
  organization={TUG},
  year=         {1988},
  note=         {Il file \texttt{btxdoc.dvi} contenente questa
                documentazione generalmente è già presente nella
                cartella \texttt{/texmf\slash doc\slash bibtex/}
                poiché fa parte integrante del sistema \TeX}
}

@book{book:latexe,
  author=       {Leslie Lamport},
  title=        {\LaTeX, a document preparation system ---
                User's guide and reference manual},
  publisher=    {Addison--Wesley Publ. Co.},
  address=      {Reading, Mass.},
  year=         {1994},
  edition=      {2}
}
```

Come si vede dai tre esempi di voci bibliografiche riportati sopra, la tipologia del documento viene sempre preceduta dal segno `@` e il contenuto della voce è racchiuso fra parentesi graffe. Il primo elemento contenuto fra queste parentesi è la chiave di citazione; nel file LaTeX, quindi, per esempio, le norme ISO per le citazioni bibliografiche *devono* essere citate con `\cite{man:refbib}` e non con un'altra chiave che piaccia di più. Conviene moltissimo comporre le chiavi mediante un prefisso, tipo 'libro', 'manuale', 'articolo', eccetera (meglio se abbreviati in modo univoco), separato dal nome 'proprio' dell'opera citata mediante, per esempio, due punti, come si può vedere negli esempi qui sopra. Il nome proprio può essere formato dal cognome del primo autore seguito da altre informazioni; si può usare qualunque carattere tranne la virgola, ma è meglio evitare gli spazi. Sono cose già dette, ma conviene ripeterle.

Le altre informazioni relative al documento citato sono specificate mediante una espressione del tipo:

`{`⟨*parola-chiave*⟩`} = {`⟨*testo*⟩`},`

dove l'ultima virgola può essere omessa solo nell'espressione finale. Ciò che nel `{`⟨*testo*⟩`}` compare fra graffe non subisce modificazioni quando viene trascritto nel file di uscita; altrimenti i programmi spesso cambiano le maiuscole in minuscole con il risultato di alterare quello che si vuole scrivere, oppure di cambiare l'ortografia di macro che verranno poi interpretate da LaTeX.

Le norme ISO sono classificate come `manual`; questa tipologia non ha bisogno dell'indicazione dell'autore; per consentire un qualunque ordinamento, allora, è opportuno inserire una parola chiave `key` che serve come chiave di ordinamento in mancanza di altre chiavi predefinite per questo scopo.

Anche la documentazione di BibTeX è classificata come manuale, ma l'autore è noto ed indicato, per cui non è necessaria una chiave supplementare.

L'ordine in cui vengono scritte le `{`⟨*parole-chiave*⟩`}` e le relative espressioni non è essenziale; l'importate è che ci siano tutte quelle relative ad una data tipologia ed eventualmente ci siano quelle facoltative che possono agevolare l'ordinamento secondo qualche criterio dipendente dallo stile bibliografico prescelto.

11.3 Programmi e pacchetti per la bibliografia

Una volta creato un database bibliografico, possiamo servirci di un programma esterno come BibTeX per estrarre le informazioni contenute nel database e generare un file adatto a LaTeXper comporre la bibliografia nel documento finale.

Il programma BibTeX può essere utilizzato anche da solo, servendosi degli stili bibliografici predefiniti di LaTeX, che inseriscono le citazioni bibliografiche con uno stile numerico, vale a dire che viene inserito un numero progressivo tra parentesi, che rimanda alla bibliografia finale, come avviene in questa guida. Utilizzando un pacchetto aggiuntivo, per esempio *natbib*, è possibile utilizzare anche altri stili, come quello "autore–data".

Il programma biber, invece, è pensato per essere utilizzato con il pacchetto *biblatex*: assieme, permettono di ottenere molti più stili bibliografici, decisamente più complessi di quelli ottenibili con BibTeX e *natbib*, oltre ad avere una migliore gestione delle lingue. Inoltre, biber ha il grande vantaggio di gestire nativamente la codifica UNICODE dei caratteri.

Vediamo dunque come ottenere delle bellissime bibliografie servendosi di questi strumenti.

11.4 BibTeX

Il programma BibTeX, fu il primo programma di questo genere distribuito con il sistema TeXquando LaTeX 208 nacque: quando lo si esegue, legge un file ausiliario `.aux` prodotto da LaTeX, assieme alle informazioni contenute in uno o più database `.bib`, e infine restituisce un file `.bbl` contenente i riferimenti bibliografici formattati (secondo lo stile prescelto), che possono essere utilizzati da LaTeX. Tutte queste operazioni avvengono grazie all'interazione tra BibTeXe alcuni comandi di LaTeX, che vanno inseriti nel documento.

11.4.1 Come comporre la bibliografia

Si inserisce il comando `\bibliography` con la seguente sintassi nel punto dove si vuole comporre la bibliografia

```
\bibliography{⟨database bibliografico⟩}
```

dove {⟨*database bibliografico*⟩} è il nome del database bibliografico che si intende usare; disponendo di più database e dovendo citare opere elencate in diversi database, il {⟨*database bibliografico*⟩} può consistere in una lista di nomi separati da virgole.

11.4.2 Come specificare lo stile bibliografico

Nel preambolo del documento bisogna specificare lo stile compositivo dell'elenco bibliografico; lo si fa con il comando `\bibliographystyle` secondo la seguente grammatica:

```
\bibliographystyle{⟨stile bibliografico⟩}
```

Eventualmente questo comando può essere preceduto dalla specificazione di un pacchetto che interagisce con la composizione della bibliografia mettendo a disposizione diverse varianti dei comandi di citazione, per esempio il pacchetto *natbib*.

Gli stili bibliografici distribuiti di default sono file con l'estensione `.bst` e si trovano nella cartella `.../bibtex/bst/` dove sono raccolti in diverse sottocartelle. Gli stili più diffusi sono `unsrt.bst` e `plain.bst`; il primo non esegue nessun ordinamento, ma le opere sono elencate nello stesso ordine in cui vengono citate; con il secondo stile le opere vengono ordinate secondo l'ordine alfabetico del primo o unico autore, o quanto al programma possa sembrare il nome di un autore, poi per anno e poi per ordine alfabetico del titolo. Con questi stili le opere vengono citate per numero d'ordine racchiuso fra parentesi quadre, come si fa abitualmente nei lavori tecnico–scientifici. Per citazioni diverse bisogna affidarsi al pacchetto di estensione *natbib* alla cui documentazione si rinvia per i dettagli necessari.

Tutti questi stili ipotizzano che la bibliografia sia scritta in inglese; non è difficile crearsi degli stili personalizzati, in particolare in italiano, ricorrendo al file `makebst.tex` che provvede a tutto il necessario pur di rispondere 'correttamente' ad un certo numero di domande; questo file va elaborato con LaTeX e richiede che si modifichi un file modello dal nome `merlin.mbs`; tutta l'operazione non è difficile, ma richiede che si sia già acquisita una certa dimestichezza con i database bibliografici e con i vari stili bibliografici.

11.4.3 Chiavi e citazioni

In certe circostanze sarebbe desiderabile introdurre nell'elenco bibliografico anche opere non espressamente citate nel testo, al limite tutte le opere contenute nel database bibliografico. Per ottenere questo scopo all'interno del documento (cioè *non* nel preambolo) si può usare il comando `\nocite` con la seguente sintassi:

```
\nocite{⟨elenco delle chiavi⟩}
oppure
\nocite{*}
```

dove ⟨*elenco delle chiavi*⟩ è una lista di una o più chiavi di citazione di opere, presenti nel database bibliografico, che si vogliono elencare nell'elenco bibliografico anche se non sono state usate come argomenti del comando `\cite` all'interno del documento. Se invece dell'elenco delle chiavi si usa l'asterisco, allora nell'elenco bibliografico compariranno tutte le opere contenute nel database bibliografico; esistono delle circostanze in cui vale la pena di comporre un database apposta (traendo le informazioni da altri database mediante la nota tecnica del "taglia e incolla") in modo da creare nel documento composto una bibliografia di consultazione, non di riferimento come solitamente avviene quando le opere elencate vengono citate nel corpo del documento.

La cosa non finisce qui; per avere la bibliografia composta, il programma deve disporre delle chiavi di citazione; perciò bisogna lanciare LaTeX. Appena ha finito e non ci sono errori, tranne eventualmente avvisi che alcune chiavi di citazione non sono state 'risolte' (il che significa che LaTeX non ha trovato altro che le chiavi, cioè gli argomenti dei comandi `\cite`, ma non sa a che cosa corrispondano queste chiavi), bisogna passare alla fase successiva.

A questo punto bisogna lanciare BibTeX o da una finestra comandi o terminal o console, oppure bisogna cliccare sull'apposito bottone o icona dello *shell editor*. In questo modo BibTeX apre il o i database specificati, cerca le voci corrispondenti alle opere citate in base alle chiavi raccolte da LaTeX nel giro precedente; le ordina secondo i criteri di ordinamento specificati nello stile bibliografico e finalmente genera un file con estensione `.bbl` che contiene in linguaggio LaTeX tutto quello che occorre per comporre la bibliografia.

Finalmente si lancia nuovamente LaTeX e questa volta viene composta la bibliografia e vengono decifrate le chiavi; ma non è finita; le chiavi sono decifrate, ma non sono ancora state usate per le citazioni; bisogna lanciare ancora una volta LaTeX in modo che possa usare le chiavi decifrate al giro precedente così che le informazioni ci siano tutte e siano state usate tutte al posto giusto. Una ulteriore passata del documento a LaTeX non guasta; serve per giustificare correttamente il documento e può rendere corretti i riferimenti incrociati che avrebbero potuto risultare modificati dall'inserimento nel testo del 'significato' delle chiavi.

Esercizio 11.1 Perché dopo la seconda volta che si lancia LaTeX dopo aver lanciato BibTeX i riferimenti incrociati potrebbero non essere corretti? Perché quindi è necessario lanciare LaTeX ancora una volta?

In sostanza la generazione e composizione della bibliografia avviene in quattro o cinque fasi così:

(1) LaTeX (2) BibTeX (3) LaTeX (4) LaTeX (5) LaTeX

Non è il caso di spaventarsi per questa (apparente) complicazione; bisogna ricordare prima di tutto che un buono *shell editor* dispone quasi sempre di un bottone o icona da cliccare che esegue tutte le passate che occorrono per avere la bibliografia composta correttamente e i riferimenti incrociati fra loro perfettamente coerenti; a cosa servirebbe sennò un buono *shell editor*?

Nello stesso tempo, se lo *shell editor* non disponesse di quel magico bottone e fosse necessario procedere a mano attraverso la finestra comandi o il terminal o la console, eseguire quei cinque comandi richiede una manciata di secondi; la compilazione di un testo di molte centinaia di pagine, diciamo 700, richiede con i PC o laptop moderni poche decine di secondi, ma evidentemente un testo non così corposo richiede un tempo minore; BibTeX richiede qualche secondo; l'intera operazione da eseguirsi una volta sola alla fine della lavorazione del documento può richiedere al massimo un paio di minuti; siccome anche l'indice analitico potrebbe richiedere lo stesso numero di passate, si può ottimizzare l'operazione prendendo due piccioni con una fava. Non è terribile.

11.5 Il pacchetto *biblatex*

Il pacchetto *biblatex* è nato per ampliare le funzioni bibliografiche di LaTeX; esso funziona assieme al programma biber, il quale si occupa di estrarre le informazioni da un database `.bib`, dello smistamento e ordinamento dei riferimenti, della generazione delle etichette necessarie, e di molte altre cose. A differenza di BibTeX, la formattazione della bibliografia avviene utilizzando solamente i comandi di LaTeX, senza far ricorso a un linguaggio esterno come avviene con i file `.bst`. Il pacchetto permette di comporre la bibliografia in modi anche molto complessi e tra le sue caratteristiche possiamo ricordare: il pieno supporto della codifica Unicode, la completa integrazione con i pacchetti *babel* e *polyglossia* per la localizzazione, la possibilità di personalizzare gli ordinamenti, la capacità di suddividere le bibliografie in più sezioni, la capacità di inserire bibliografie multiple nello stesso documento, la modifica dinamica dei dati. Questa è solo una lista parziale, le funzioni sono davvero numerose e bisogna riferirsi al manuale di *biblatex* per sapere quali usare e come usarle.

Per servirsi di tutte queste belle possibilità offerte da *biblatex* è necessario comporre il nostro documento avendo cura di inserire alcune specificazioni nel preambolo, simili a quelle che seguono:

```
\usepackage[⟨lista lingue secondarie,⟩⟨lingua principale⟩.main]{babel}
\usepackage[autosyle]{csquotes}
\usepackage[backend=biber,hyperref,style=⟨nome dello stile⟩]{biblatex}
\addbibresource{⟨database⟩.bib}
```

Il pacchetto *babel* dovrebbe essere già caricato, ma lo si ricorda perché è necessario per permettere a *biblatex* di comporre la bibliografia con la giusta localizzazione. Anche *polyglossia* è pienamente supportato; se lo si vuole usare, i comandi vanno modificati opportunamente, come si vede nella pagina 139.

Il pacchetto *csquotes* serve per usare le virgolette uncinate (caporali) nelle citazioni. Quando si carica il pacchetto *biblatex* va sempre specificata l'opzione *style*: lo stile bibliografico definisce quante e quali informazioni devono apparire nella citazione bibliografica, e stabilisce come deve apparire la bibliografia; il pacchetto è provvisto internamente di cinque stili per la bibliografia, con numerose

variazioni per le citazioni, più tre stili "speciali" riservati a usi particolari; se ancora non fosse sufficiente, con l'installazione di TEX Livesono già inclusi più di una cinquantina di di file esterni che definiscono ulteriori stili, che implementano alcuni di quelli più diffusi, come lo Harvard o il Chicago, o quelli di importanti editori scientifici, assieme ad altri stili specialistici. Le opzioni vanno tipicamente espresse nella forma *chiave* = *valore*: le possibilità sono davvero molte, ancora una volta è fondamentale la lettura del manuale del pacchetto. L'opzione *hyperref* serve per comporre correttamente i collegamenti ipertestuali; l'opzione *backend*=`biber` specifica che bisogna usare biber come estrattore bibliografico; è possibile utilizzare anche BIBTEX, ma; salvo ragioni molto particolari, non è consigliato. Il comando `\addbibresource` indica quale database bibliografico si intende usare; nel caso siano più di uno è sufficiente ripetere il comando.

Infine, va inserito nel punto in cui si desidera comporre la bibliografia, il comando

```
\printbibliography
```

Prima del 2012 l'estrattore bibliografico da usare con *biblatex* era BIBTEX; dal 2012 è biber che è nato proprio per superare le limitazioni di BIBTEX. Il vantaggio principale è quello che può gestire molto meglio i caratteri codificati in UNICODE, cosa che diventa importantissima quando si debbano comporre bibliografie che contengano dei riferimenti in lingue scritte con alfabeti diversi da quello latino. Il pacchetto *biblatex* può funzionare con entrambi i programmi per ragioni di retro-compatibilità e la documentazione marca in modo chiaro quali prestazioni non siano ottenibili con BIBTEX. Si consiglia comunque di usare solo biber e di configurare lo *shell editor* affinché esso usi biber quando gli si chiede di comporre una bibliografia. Invece di riconfigurare lo shell editor, si può anche usare la 'riga magica'

```
%␣!BIB␣TS-program␣=␣biber
```

se lo shell editor è in grado di interpretare le righe magiche.

Il modello di database usato da *biblatex* è ereditato da quello di BIBTEX; la maggior parte dei campi è uguale e c'è una buona compatibilità tra i due. Nel manuale di *biblatex* sono indicate le principali differenze:

- il tipo di materiale `inbook`;
- i campi `institution`, `organization` e `publisher`;
- alcuni tipi di titoli;
- il campo `series`;
- i campi `year` e `month`, che si possono ancora usare, ma è preferibile utilizzare il nuovo campo `date`, che è molto più versatile;
- il campo `edition`;
- il campo `key`.

Inoltre, il modello usato da *biblatex* è più versatile e introduce numerosi campi nuovi, oltre a consentire l'inserimento di campi personalizzati, che possono

essere utilizzati dai pacchetti di stile esterni per consentire delle bibliografie più elaborate, come, ad esempio, quelle in uso in ambito giuridico.

La versatilità del pacchetto *biblatex* si dimostra anche nella capacità di mettere a disposizione alcuni comandi per richiamare le informazioni bibliografiche estratte da biber, come, ad esempio i comandi `\citeauthor`, `\citetitle` o `\citedate`. Inoltre, sono presenti anche moltissime varianti del comando `\cite`, per adattarsi meglio ai vari contesti dove potrebbe essere inserito un riferimento bibliografico, come, ad esempio, un riferimento tra parentesi, o in una nota, o come soggetto di una frase. Vediamo una breve lista di alcuni di essi:

`\parencite` inserisce la citazione tra parentesi, di norma tonde, oppure quadrate per gli stili numerici e alfabetici;
`\footcite` inserisce la citazione in nota a piè di pagina;
`\smartcite` agisce come `\parencite` in una nota, e come `\footnote` nel testo;
`\textcite` da usare quando si vuole fare un riferimento all'interno del discorso, il comando inserisce il nome dell'autore con un breve riferimento tra parentesi

Alcuni comandi sono generici, altri sono dipendenti dallo stile, e nel caso di stili esterni, il loro comportamento può essere anche ridefinito, per cui si raccomanda di consultare sia la guida di *biblatex*, dove sono illustrati molti altri comandi per le citazioni, sia la guida dell'eventuale stile esterno scelto, per capire come sfruttare al meglio tutte le funzionalità messe a disposizione.

11.6 Quanti database bibliografici?

È chiaro che questa è una domanda insolita e alla quale è difficile dare una risposta. Già prima si è accennato alla possibilità di creare database bibliografici divisi per argomenti; questa è già una risposta parziale, ma bisogna associarla al tipo di documento che si vuole comporre.

In un articolo o rapporto scientifico è possibile che si debbano riportare solo le voci bibliografiche dei riferimenti citati; perciò anche un unico database generale può essere utile, come potrebbero esserlo diversi database specialistici. Invece un testo di tipo illustrativo[2] è probabile che l'elenco contenga anche opere non citate nel testo, ma venga usato il comando `\nocite{*}` per comporre nella bibliografia l'intero database bibliografico.

Il modo migliore di gestire più database è quello di affidarsi a software specifici, chiamati *reference manager*; oltre a JabRef e BibDesk, che abbiamo già nominato nell'introduzione di questo capitolo, vale la pena segnalare anche Zotero, da scaricare dalla rete, che con la sua estensione 'Better BibTeX' svolge un ottimo lavoro. All'interno del programma è sufficiente creare delle cartelle dedicate

[2] In certe università si usa l'orrenda parola *compilativo*, a cui è spesso associata una connotazione negativa – se il testo raccoglie solo titoli di articoli e di riviste, effettivamente serve a poco, ma se si tratta di una raccolta ragionata, commentata, verificata, di opere che descrivono davvero *lo stato dell'arte* allora il testo diventa tutt'altro che spregevole, anzi è una benedizione!

ciascuna a un argomento, a una disciplina, o allo specifico documento a cui si sta lavorando, e con un semplice comando verrà creato un database con tutti i riferimenti di quella cartella. Se si vuole comporre un documento informativo dove si vogliono riportare tutti i riferimenti di un database bibliografico, anche quelli non citati, il comando `\nocite{*}` e BibTeX o *biblatex* con biber fanno il resto. Il programma inoltre è fornito di strumenti di de-duplicazione dei record, quindi può verificare al posto nostro e in maniera più efficace che ciascun riferimento compaia una sola volta all'interno dei vari database, e che la chiave di citazione sia davvero univoca.

Se si sta scrivendo un documento dove la bibliografia deve contenere solamente i riferimenti citati nel testo, si può procedere in questo modo: durante la scrittura si tengono aperti anche i file dei database bibliografici al fine di riportare le chiavi di chiamata semplicemente copiandole da ciascun database (senza perdere l'occasione di verificare che gli elementi necessari all'identificazione dell'opera siano presenti, non debbano essere aggiornati, e cose di questo genere); se per questa operazione bisogna aprire un altro database, si coglie l'occasione per trascriverne il nome nell'elenco di database passati al comando `\bibliography` (per BibTeX) o a `\addbibresource` (per *biblatex*). Nel documento *non* va usato il comando `\nocite{*}`, ma si lascia al programma di estrazione bibliografica il compito di recuperare dai database solo tutti i riferimenti citati.

In molti casi è pratico anche tenere aggiornato un unico database, mettendolo a disposizione di più documenti. Uno strumento a disposizione è quello dei link simbolici, disponibili anche sulle macchine Windows a partire da Windows Vista. Il link simbolico fa credere al sistema operativo che nella cartella dove c'è il nome del link simbolico ci sia davvero il file originale. In questo modo l'aggiornamento o la correzione del *file vero* esistente in un'unica copia, si riflette automaticamente su tutti i documenti che hanno fatto riferimento a quel database tramite il link simbolico. Un programma di estrazione bibliografica cerca i database nella stessa cartella dove si trova il file principale del documento che si sta componendo, quindi nella cartella che contiene il file principale si aggiungono i link simbolici ai database che si trovano altrove. Per le macchine di tipo UNIX (Linux, Mac, eccetera) si userà la sintassi:

```
ln -s {⟨percorso completo del file vero⟩} {⟨nome del link simbolico⟩}
```

mentre su una macchina Windows con sistema operativo Vista o successivo, il comando diventa:

```
mklink {⟨nome del link simbolico⟩} {⟨percorso completo del file vero⟩}
```

Solitamente si usa come {⟨*nome del link simbolico*⟩} il nome del *file vero* privato di tutto il percorso che vi conduce, di cartella in cartella, a partire dal nome del disco fisso.

È incredibile quanto tempo si risparmia a seguire le indicazioni riportate sopra. Ma è anche incredibile quanto risultino composte bene le bibliografie con

questi strumenti; s'intende che i database bibliografici non devono contenere errori – questo va da sé!

Capitolo 12

LaTeX: indici e glossari

12.1 Introduzione

Gli indici analitici e i glossari possono essere molto utili nelle opere di consultazione. LaTeX offre un grande aiuto per la compilazione di questi elenchi, ma la parte difficile resta quella dell'autore che deve decidere che cosa inviare all'indice analitico o al glossario, se deve usare voci indipendenti o subordinate, se deve distinguere il modo di scrivere le pagine a seconda di come il lemma di quella voce viene trattato; definizione, esempio d'uso, applicazione, eccetera.

12.2 L'indice analitico

L'indice analitico è un elenco di voci ordinate alfabeticamente anche con alcuni livelli di subordinazione, e vicino ad ogni lemma viene inserito il numero della pagina dove quel lemma viene trattato. Il lemma può essere trattato in diversi punti del testo, quindi di fianco al lemma può comparire un elenco di numeri di pagina.

La raccolta di queste informazioni viene fatta mediante il comando `\index`; la sintassi è

`\index{`⟨*lemma*⟩`}`

Se nel preambolo c'è l'istruzione `\makeindex`, l'informazione specifica relativa al ⟨*lemma*⟩ viene inviata ad un file ausiliario da elaborare in un secondo tempo; se quell'istruzione manca, nel file ausiliario non viene scritto niente. Quando i calcolatori erano lenti, era importante non rallentare l'elaborazione del file sorgente con operazioni non strettamente necessarie; oggi la cosa non è più così importante, ma visto che l'indice analitico è l'ultima cosa che si compone in un libro, è bene non perdere tempo con questa informazione finché non si è alla fine o quasi alla fine.

L'informazione che viene scritta da LaTeX nel file ausiliario (il cui nome ha l'estensione `.idx`) è del tipo:

`\indexentry{`⟨*lemma*⟩`}{`⟨*pagina*⟩`}`

Per ottenere questa raccolta di dati basta collocare l'istruzione `\index` con il suo argomento 'attaccata', cioè senza lasciare spazi, alla parola che si vuole indicizzare. Per esempio:

```
... la tecnologia dei transistori\index{transistore}
      NPN\index{transistore!NPN} ...
```

12.2.1 Il programma makeindex

Per elaborare il file `.idx` in modo da metterlo nella forma giusta, cioè in ordine alfabetico, strutturato per lemmi, sottolemmi e sotto sottolemmi, con gli elenchi di numeri di pagine senza doppioni o con intervalli, eventualmente composti con caratteri diversi a seconda dell'uso del lemma, esiste il programma makeindex; si tratta di un programma da lanciare a mano dalla finestra comandi o dal terminal o dalla console oppure premendo l'opportuno pulsante dello *shell editor*.

Questo programma esamina il ⟨*lemma*⟩ come trascritto senza modifiche dall'argomento di `\index` nel file `.idx` e lo elabora secondo quanto vi trova indicato.

Infatti quello che appare in ⟨*lemma*⟩ non è necessariamente il solo 'lemma' ma anche informazioni ausiliarie che facilitano il compito del programma makeindex.

Nel primo esempio elementare indicato sopra, il ⟨*lemma*⟩ è costituito semplicemente dalla parola 'transistore'; questa parola serve a diversi scopi.

1. Indica che si tratta di un lemma di primo livello;
2. Indica che la chiave con cui eseguire l'ordinamento alfabetico è 'transistore';
3. Indica che va riportato nell'indice analitico tale e quale, vale a dire stampato con il font di default con cui viene composto l'indice;
4. Indica che il numero della pagina in cui compare va scritto anch'esso con il font di default.

Strutturando il ⟨*lemma*⟩ in modo più articolato le varie funzioni indicate nella precedente enumerazione possono essere trattate in modo diverso così da far apparire il livello del 'lemma' nell'indice, per indicare una diversa chiave di ordinamento, per indicare come scrivere il 'lemma' nell'indice e come stampare il numero della pagina; la strutturazione consente anche di indicare l'inizio e la fine degli intervalli di pagine, per inserire nell'indice il rinvio ad un altro lemma, eccetera.

I dettagli sono spiegati bene nella documentazione del programma makeindex che si trova nella cartella `.../doc/support/makeindex/` dove non solo è contenuto il file `makeindex.pdf` ma anche un semplice tutorial `ind.pdf`.

Sommariamente qui si riportano due esempi:

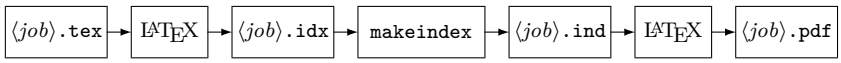


Figura 12.1: Le fasi per la produzione dell'indice analitico

1. Scrivendo

   ```
   \index{transistore!NPN}
   ```

   il lemma 'NPN' viene scritto strutturato sotto il lemma 'transistore'.

2. Scrivendo

   ```
   \index{ambiente!enumerate@\textsf{enumerate}}
   ```

   il lemma 'enumerate' viene scritto con il font senza grazie e viene strutturato sotto il lemma 'ambiente'.

Naturalmente per sfruttare tutta la forza del programma makeindex bisogna leggere attentamente la documentazione citata e prendere esempio dal tutorial menzionato sopra.

12.2.2 La composizione effettiva dell'indice analitico

La composizione effettiva dell'indice analitico avviene specificando nel file sorgente per LaTeX:

```
\input{⟨file⟩.ind}
```

dove ⟨*file*⟩ è il nome del 'job' e `.ind` ne è l'estensione; il 'job' coincide con il nome del file sorgente per LaTeX che ha dato luogo alla raccolta dei dati per l'esecuzione dell'indice analitico ed è stato elaborato da `makeindex` secondo lo schema della figura 12.1. Questo file contiene anche i comandi di apertura e di chiusura dell'ambiente *theindex*, quindi se non si vuole altro, basta semplicemente eseguire l'inclusione di questo file tramite il comando `\input`.

Come la figura dimostra chiaramente, per ottenere la composizione definitiva dell'indice analitico bisogna lanciare LaTeX ancora una volta, perché l'indice non può essere composto finché il file ⟨*job*⟩`.ind` non è disponibile (vedi però più avanti).

Usando il pacchetto di estensione *makeidx* è possibile servirsi del semplice comando `\printindex` che provvede da solo a leggere il file `.ind` e a comporre l'indice analitico.

Siccome i numeri delle pagine devono essere quelli definitivi, è evidente che la composizione dell'indice analitico deve essere in assoluto l'ultima operazione da fare; se si deve eseguire una qualche correzione nel testo, questa correzione comporta anche la correzione dell'indice analitico, vale a dire la duplice esecuzione

di LaTeX con l'esecuzione di makeindex dopo la prima di LaTeX. Per fortuna gli *shell editor* hanno generalmente un bottone solo che esegue tutta questa procedura, necessaria anche per comporre la bibliografia usando BibTeX o biber.

12.3 Il glossario

Un glossario è un elenco di voci accompagnate da una breve spiegazione; talvolta appare anche la pagina del testo dove viene riportata una spiegazione meno succinta.

Il procedimento per produrre un glossario è simile a quello usato per comporre l'indice analitico, ma è un po' più artigianale, come si vedrà fra poco.

L'inizio consiste nel marcare il testo sorgente in una maniera simile a quella dell'indice analitico:

```
\glossary{⟨voce⟩}
```

Lanciando LaTeX, se il preambolo contiene l'istruzione `\makeglossary`, viene prodotto anche un file ⟨*job*⟩`.glo` che contiene tutte le voci di cui si vuol generare il glossario accompagnate dal numero della pagina dove esse compaiono insieme ad una loro spiegazione non succinta.

Lanciando il programma makeindex dalla linea di comando

```
makeindex -s gglo.ist -o ⟨job⟩.gls ⟨job⟩.glo
```

si produce l'elenco alfabetizzato ⟨*job*⟩`.gls` delle parole di cui si desidera comporre il glossario;

Bisogna aprire questo file e aggiungere la breve spiegazione oltre che a correggere la forma di stampa del numero di pagina (per esempio per cancellarlo, se esso non svolge nessuna funzione specifica, come quando il glossario non è altro che un elenco di simboli o di abbreviazioni); poi bisogna comporlo in una maniera simile a quello che si fa per la composizione dell'indice analitico.

In questo senso la composizione del glossario è un po' più artigianale, in quanto bisogna intervenire ancora sul file alfabetizzato.

Esistono diversi pacchetti in rete per comporre glossari o simili elenchi commentati. Qui si suggerisce il pacchetto *glossaries* già presente in ogni installazione di TeX Live completa; esso offre una maniera comoda per scrivere la ⟨*voce*⟩ in modo che contenga direttamente anche la descrizione della voce, così che quando si compone il file alfabetizzato esso sia già pronto per essere composto con LaTeX. Il pacchetto, ovviamente, definisce il comando `\printglossary`, del tutto simile nelle sue funzioni a `\printindex`.

Si raccomanda la lettura attenta della documentazione del pacchetto *glossaries* perché esso offre diverse altre funzioni avanzate che possono tornare molto comode per la generazione di un glossario e per l'amministrazione delle voci da inserirvi.

Il manuale del pacchetto si legge con il solito comando da terminale o prompt dei comandi `texdoc glossaries`; siccome il pacchetto svolge moltissime funzioni, il manuale è piuttosto esteso; ma l'autrice Nicola Talbot ha scritto un'altra guida introduttiva, specialmente adatta ai nuovi utenti; la si legge con il comando `texdoc glossariesbegin`. Siccome *glossaries* rimpiazza il precedente e ormai obsoleto pacchetto *glossary*, l'autrice ha anche predisposto una terza guida per migrare precedenti documenti al nuovo pacchetto; la si legge con `texdoc glossary2glossaries`. È quanto mai opportuno riferirsi a queste guide, perché il pacchetto è molto versatile e senza queste guide non lo si può usare efficientemente.

12.4 Come modificare la composizione dell'indice analitico

Quanto viene esposto qui vale anche per modificare la composizione del glossario.

Per entrambi il file che compone l'elenco richiesto è prodotto dall'esecuzione del programma makeindex; per l'indice analitico viene prodotto il file ⟨*job*⟩`.ind` mentre per il glossario viene prodotto il file ⟨*job*⟩`.gls`. Entrambi questi file racchiudono l'elenco da comporre dentro due specifici ambienti *theindex* e *theglossary*. Come ogni voce sia articolata dipende da come si siano usati i comandi `\index` o `\glossary`, e quindi da come le varie opzioni siano state elaborate da makeindex per produrre i file di uscita. Cambiando opzioni, chiaramente, cambia anche la composizione.

Tuttavia qui si vuole richiamare l'attenzione su un altro punto e per esemplificare ci riferiremo all'indice analitico; per il glossario valgono considerazioni del tutto analoghe.

La formattazione dell'indice analitico dipende da come è definito l'ambiente *theindex* oltre che dal contenuto del file ⟨*job*⟩`.ind`. Cambiare formattazione dell'indice analitico, implica la capacità di cambiare la definizione dell'ambiente *theindex*. Si tratta di modificare la definizione di un ambiente di sistema e l'operazione, benché non difficile, richiede un minimo di attenzione e di conoscenza. L'argomento viene rimandato al paragrafo 19.9 nel capitolo 19.

12.5 La creazione sincrona dell'indice analitico e/o del glossario

Si è già detto altrove che le versioni moderne delle distribuzioni del sistema TEX usano come interpreti i programmi pdftex, luatex e xetex ai quali viene associato un determinato file di formato, nel nostro caso il formato che contiene le definizioni del mark up del linguaggio LATEX; a seconda di quali opzioni siano attive al momento della inizializzazione per la creazione del formato e/o al momento della compilazione, l'interprete può svolgere determinate azioni, in particolare può

Solo pdflatex

produrre in uscita il file `.dvi` oppure il file `.pdf`; può eseguire i comandi estesi del programma etex, oppure può riferirsi solo ai comandi dell'interprete tex; può chiedere al sistema di eseguire comandi esterni, oppure di non farlo.

In questo paragrafo viene esposta una soluzione al problema di eseguire immediatamente la conversione del file grezzo dell'indice analitico `.idx` nel file indicizzato e, volendo, la conversione del file grezzo del glossario `.glo` nel file indicizzato `.gls`. Questa operazione è possibile se l'interprete che si sta usando è abilitato per l'esecuzione di comandi di sistema.

Generalmente parlando questa operazione può venire eseguita a mano, in differita, cliccando sull'apposito bottone dello *shell editor* con cui si sta eseguendo la lavorazione del documento; normalmente non ci sono problemi, mentre per il glossario è necessario esprimere diverse opzioni al comando di sistema, per cui l'operazione, benché non sia difficile da eseguire dalla finestra comandi, come è stato descritto nei paragrafi precedenti, resta comunque una operazione suscettibile di errori. Per l'indice analitico l'operazione attraverso la finestra comandi diventa necessaria se si vogliono usare delle opzioni specifiche per il comando makeindex.

Ricordiamo innanzi tutto che l'indice analitico e il glossario vengono composti per ultimi, quando non è più necessario raccogliere informazioni da includere dentro questi elenchi. Ricordiamo anche che entrambi i file indicizzati hanno il nome identico a quello del file principale del documento che si sta componendo ⟨*job*⟩, e che questo nome è contenuto nella macro `\jobname`.

Allora è possibile eseguire la trasformazione immediata dei due file ⟨*job*⟩`.idx` e ⟨*job*⟩`.glo` subito dopo averli completati e subito prima di comporre ciascuna lista indicizzata ⟨*job*⟩`.ind` e ⟨*job*⟩`.gls`. Questo implica che bisogna chiudere immediatamente i due flussi di scrittura dell'interprete verso i file ⟨*job*⟩`.idx` e, rispettivamente, ⟨*job*⟩`.glo`. I nomi di questi due flussi di scrittura sono contenuti rispettivamente nelle due macro interne e protette di LaTeX `\@indexfile` e `\@glossaryfile`[1]. La chiusura di questi due flussi di scrittura deve essere istantanea, non differita alla prima occasione opportuna, come avviene invece normalmente per la scrittura su altri file.

Lo scrivente è ricorso all'uso del pacchetto *imakeidx* che ricorre al comando `\write18`; questo a sua volta permette di chiedere al sistema di eseguire l'argomento di questo comando, come se fosse un comando scritto nella finestra comandi, terminal, xterm, console, comunque si chiami nel sistema operativo che governa il calcolatore che si sta usando.

[1] Talvolta è necessario comporre indici analitici o glossari tematici; alcune classi non standard come *memoir* hanno un meccanismo interno che *non* consente di agire come descritto in questo e nei prossimi paragrafi. Altre classi non standard, come per esempio quelle della collezione KOMA-script, invece, generano inizialmente un solo file `.idx`, ma poi, grazie ad un applicativo esterno, spezzano questo file in un certo numero di file grezzi, con nomi distinti, ognuno dei quali va elaborato da makeindex. È possibile scrivere una serie di macro per gestire queste situazioni, e lasciare che i comandi descritti in questo e nei prossimi paragrafi, eseguano tutto il lavoro per ciascuno degli indici o glossari. È appunto quello che si è fatto per i vari indici analitici di questo testo, sebbene la classe usata per la sua composizione sia la classe standard *book*.

È bene notare che i programmi di composizione pdftex, lualatex e xelatex, pur potendo eseguire il comando `\write18`, di default ne erano disabilitati. Si tratta di sicurezza informatica. Ma se talvolta per scopi specifici, l'utente vuole ricorrere a questa funzionalità, doveva espressamente abilitarla. Per questo scopo era necessario scrivere l'apposita opzione nella linea di comando da inserire a mano da una finestra comandi, o da inserire nella configurazione dello *shell editor*. Per gli eseguibili della distribuzione MiKTEX l'opzione da inserire nella linea di comando è `--enable-write18`; per gli eseguibili della distribuzione TEX Live l'opzione è `-shell-escape`.

Per fortuna dal 2013 non è più necessario ricorrere a questa abilitazione perché l'autorizzazione ad eseguire lo *shell escape* è già automatica per un piccolo numero di programmi 'sicuri' della distribuzione del sistema TEX; fra questi c'è makeindex (ma non c'è xindy, vedi più avanti).

12.6 Composizione automatica dell'indice analitico

I pacchetti *imakeidx* e *indextools* permettono di comporre gli indici analitici in modo 'sincrono' senza bisogno di fare nulla se non specificare qualche impostazione iniziale. Sono stati scritti per comporre gli indici analitici, ma si ritiene che siano usabili anche per i glossari; chi scrive non li ha mai provati in questo senso.

Caricato l'uno o l'altro pacchetto (qui si esemplifica con *imakeidx* ma quanto detto vale tale e quale con *indextools* che non è altro che un *fork* del precedente) nel solito modo, bisogna specificare di voler comporre uno o più indici:

```
\usepackage[⟨opzioni del pacchetto⟩]{imakeidx}
\makeindex[⟨opzioni di makindex⟩]
```

Nel corpo del testo i termini da indicizzare vengono introdotti con

```
\index[⟨opzioni per l'indicizzazione⟩]{⟨lemma⟩}
```

Verso la fine del documento, al momento di stampare l'indice o gli indici analitici, si scrive

```
\printindex[⟨opzioni per la stampa⟩]
```

Come si vede, a parte le opzioni, i comandi da usare sono gli stessi che si userebbero senza ricorrere a questi due pacchetti di estensione.

Chiaramente le ⟨*opzioni per il pacchetto*⟩ sono globali e vanno scelte fra quelle esposte nelle rispettive documentazioni; spesso non è necessaria nessuna di queste opzioni, ma all'occorrenza esistono e sono accessibili.

Le ⟨*opzioni per makeindex*⟩ sono le più interessanti; quasi tutte sono della forma *chiave* = *valore*; qui se ne citano alcune:

name =⟨*nome*⟩ È il nome simbolico per identificare un particolare indice quando si compongono diversi indici; se non si specifica nulla viene prodotto il file grezzo `\jobname.idx`, altrimenti viene prodotto il file ⟨*nome*⟩`.idx`.

title = ⟨*titolo*⟩ dell'indice con il ⟨*nome*⟩ simbolico. Usando queste due chiavi non è difficile creare un indice identificato con `Nomi` che, quando sarà composto, invece del titolo di default "Indice analitico", avrà il titolo, per esempio, "Nomi dei personaggi importanti".

options = `{`⟨*opzioni*⟩`}` La lista delle ⟨*opzioni*⟩ da passare a programma di composizione dell'indice (di default il programma makeindex), per specificargli, per esempio, il file di stile da usare. Si noti che il 'valore' di questa opzione deve essere racchiuso fra parentesi graffe.

intoc è una chiave booleana che non necessita di specificarne il 'valore, che di default vale 'true'; bisogna specificare, eventualmente, solo il valore 'false' ma per ottenere questo risultato basta non specificare l'opzione. Serve per far sì che il titolo del particolare indice sia inserito nell'indice generale.

columns = ⟨*numero*⟩ Serve per specificare come comporre l'indice analitico in colonne; di default ⟨*numero*⟩ vale 2, ma si può specificare 1, per esempio per un indice dei nomi o per un glossario; mentre si può specificare un ⟨*numero*⟩ maggiore di 2 per indici che usano lemmi molto corti.

Sono disponibili anche altre opzioni, per le quali si rinvia alla documentazione.

Per il comando `\index` è disponibile una sola opzione, il ⟨*nome*⟩ che identifica il particolare indice in cui inserire il ⟨*lemma*⟩.

Analogamente il comando `\printindex` accetta come opzione solo il ⟨*nome*⟩ simbolico dell'indice che si vuole stampare. Se è necessario è possibile inserire fra il titolo dell'indice identificato da ⟨*nome*⟩ e l'indice vero e proprio un testo esplicativo composto a giustezza piena, non nella prima colonna dell'indice; questo testo si ottiene con il comando:

`\indexprologue[`⟨*spazio*⟩`]{`⟨*testo esplicativo*⟩`}`

Lo spazio viene inserito fra il titolo dell'indice e il ⟨*testo esplicativo*⟩. Composto l'indice, il ⟨*testo esplicativo*⟩ viene annullato, per cui se per un indice successivo occorre un altro ⟨*testo esplicativo*⟩ basta usare il comando senza dover preoccuparsi di 'cancellare' il testo precedente.

Gli indici analitici di questa guida sono stati composti usando questo metodo; anche se gli indici vengono ricomposti ogni volta che si ricompone il documento, il 'tempo perso' è complessivamente di un secondo o due, e compensa abbondantemente la necessità di ricordarsi di aggiornare l'indice analitico ogni volta che si fanno delle modifiche al testo o comunque anche quando il testo è pronto per essere licenziato e caricato nel sito GUIT.

Tanto per mostrare un esempio qui si mostrano le impostazioni con le quali si sono composti i vari indici analitici di questo testo.

1. Chiamata del pacchetto *imakeidx* (nel preambolo):

```
\usepackage{imakeidx}
```

2. Impostazione di comandi `\makeindex` (nel preambolo):

```
\makeindex[options={-s GuidaGuIT.ist},intoc]% macroistruzioni
\makeindex[options={-s GuidaGuIT.ist},intoc,
   name=GuidaGuITamb,title=Indice degli ambienti]
\makeindex[options={-s GuidaGuITfiles.ist},intoc,
   name=GuidaGuITfile,title=Indice dei file]
\makeindex[options={-s GuidaGuIT.ist},intoc,
   name=GuidaGuITclass,title=Indice delle classi]
\makeindex[options={-s GuidaGuIT.ist},intoc,
   name=GuidaGuITpack,title=Indice dei pacchetti]
\makeindex[options={-s GuidaGuIT.ist},intoc,
   name=GuidaGuITprog,title=Indice dei programmi
      e~delle distribuzioni]
```

3. Definizione di alcuni comandi con i quali contemporaneamente si scrive una parola nel testo e la si manda al suo indice[2]:

```
\DeclareRobustCommand*{\cs}[1]{% macroistruzioni
   \texttt{\char92#1}\index{#1@\texttt{\char92#1}}}
\DeclareRobustCommand*\prog[1]{%
   \progstyle{#1}\index[GuidaGuITprog]{#1@\progstyle{#1}}}
\DeclareRobustCommand*\pack[1]{%
   \packstyle{#1}\index[GuidaGuITpack]{#1@\packstyle{#1}}}
...
\DeclareRobustCommand*\file[1]{%
   \filestyle{#1}\index[GuidaGuITfile]{#1@\filestyle{#1}}}
```

   Come si vede, ogni tipologia di voce è accompagnata dalla modalità per scriverla; ovviamente i vari comandi `\progstyle`, `\packstyle`, `\filestyle`, eccetera sono stati definiti a parte, ma non si tratta di definizioni che nascondono particolari difficoltà; il risultato di queste impostazioni può essere visto lungo tutto il testo e nei vari indici analitici. Si noti come sono state trattate le macroistruzioni: la barra rovescia, indicata con l'istruzione nativa di TEX `\char92`, non viene usata nella stringa di alfabetizzazione, ma viene usata solo per scriverla nel testo e nell'indice, senza che la macroistruzione sia eseguita: in sostanza `\char92macro` e `\macro` non sono la stessa cosa, perché la prima non è una macro ma una stringa che viene composta come se fosse una macro. Questo, forse, è l'unico trucco a cui si è dovuto ricorrere per gestire le macro negli indici e nel testo.

[2]Per come si possono definire nuovi comandi si veda il capitolo 19.

4. Stampa degli indici analitici con `\printindex`:

```
\printindex % indice delle macroistruzioni
\printindex[GuidaGuITamb]
\printindex[GuidaGuITclass]
\printindex[GuidaGuITfile]
\printindex[GuidaGuITpack]
\printindex[GuidaGuITprog]
```

12.7 Conclusione

Forse si è scesi troppo nei dettagli per descrivere il funzionamento della creazione di indici analitici e glossari, ma tutti questi dettagli diventano irrilevanti quando si usino o l'uno o l'altro dei pacchetti *imakeidx* o indextools.

Sebbene uno o più indici analitici siano raramente necessari in un testo, tranne nei testi di consultazione, così come la presenza di uno o più glossari, la possibilità di comporli in automatico mediante i pacchetti indicati fa sì che questa funzionalità sia facilissima da ottenere senza doversi preoccupare della moltitudine di comandi di basso livello che sarebbe altrimenti necessario usare.

Perciò se si decide di comporre indici analitici o glossari, si usino questi pacchetti e non si perda tempo a contare quante compilazioni bisogna rifare per avere tutto a posto.

II

Volume

Capitolo 13

LaTeX: la matematica semplice

13.1 Introduzione

La forza di LaTeX, che ne ha determinato la grandissima diffusione iniziale in ambito accademico e tecnico, prima di diffondersi in ambito letterario e umanistico, è proprio la professionalità con la quale compone la matematica; in questo primo capitolo ci soffermeremo sulle strutture matematiche più semplici e nel prossimo capitolo scenderemo nei dettagli per strutture matematiche più complesse, avvalendoci del pacchetto di estensione `amsmath`.

Non è sempre necessario ricorrere al pacchetto `amsmath`; dipende dal tipo di matematica che si vuole comporre. In questo capitolo, quindi, ci familiarizziamo solamente con la composizione della matematica, ma, anche se LaTeX, così com'è appena installato, è già in grado di comporre strutture matematiche piuttosto avanzate, rimandiamo al prossimo capitolo ciò che, pur essendo componibile con LaTeX senza estensioni, riesce molto più comodo da comporre con le estensioni fornite dal pacchetto `amsmath`.

13.2 I modi matematici

LaTeX compone la matematica sostanzialmente in quattro modi; se si andasse per il sottile si scoprirebbe che i modi sono otto, ma questo il lettore interessato lo può studiare nel TeXbook. I modi che interessano la quasi totalità degli utenti sono i seguenti:

1. Il modo testuale. LaTeX compone in questo modo quando scrive una espressione matematica in linea con il testo, come quando si specifica che la sezione aurea è $\varphi = (\sqrt{5}-1)/2$ e che questo numero gode della proprietà che $1/\varphi = 1 + \varphi$. Se si vuole comporre in questo modo anche quando

LaTeX di default comporrebbe in modo diverso bisogna dare l'istruzione `\textstyle`.

2. Il modo display. LaTeX compone la matematica in display quando compone le espressioni in linee a sé stanti, staccate dal testo precedente e seguente mediante spazi bianchi di ampiezza adeguata per 'mettere in mostra' l'espressione; per esempio[1]:

$$\varphi = \frac{\sqrt{5}-1}{2} \qquad \text{e} \qquad \frac{1}{\varphi} = 1 + \varphi \qquad \text{(sezione aurea)}$$

   Se si vuole comporre in questo modo anche quando LaTeX di default comporrebbe in modo diverso bisogna dare l'istruzione `\displaystyle`.

3. Il modo degli indici primi. LaTeX compone gli apici e i pedici (gli esponenti e i deponenti) di primo ordine in questo modo; usa un carattere di corpo più piccolo rispetto a quello delle variabili principali e li rialza per gli apici o li abbassa per i pedici di una quantità costante e prestabilita dalle caratteristiche dei font usati; per esempio in display si hanno le soluzioni di una equazione di secondo grado nella forma:

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

   Se si vuole comporre in questo modo anche quando LaTeX di default comporrebbe in modo diverso bisogna dare l'istruzione `\scriptstyle`.

4. Il modo degli indici secondi. LaTeX compone gli apici e i pedici (gli esponenti e i deponenti) di secondo ordine, cioè quelli applicati agli apici e ai pedici di primo livello, con un carattere ancora più piccolo e adeguatamente rialzato o ribassato rispetto alle variabili a cui si riferiscono, che sono già per loro conto composte con il font degli indici di primo livello; per esempio:

$$V_{R_\mathrm{e}} = R_\mathrm{e} I_\mathrm{e}$$

   Se si vuole comporre in questo modo anche quando LaTeX di default comporrebbe in modo diverso bisogna dare l'istruzione `\scriptscriptstyle`.

Di solito è piuttosto raro che si debba specificare lo stile, ovvero il modo di composizione matematica; è molto meglio lasciare fare a LaTeX che sa quasi sempre qual è il modo giusto da usare per la composizione.

Per formule così semplici come quelle esposte nell'enumerazione precedente basta che il compositore scriva il testo sorgente in lettere nello stesso modo come se lo dettasse al telefono (in inglese). Per esempio:

[1] Spesso, specialmente nella letteratura anglosassone, con ϕ si intende il numero aureo $(\sqrt{5}+1)/2$. Si riesce benissimo a distinguere dal contesto del discorso a quale dei due significati sia associato il simbolo; qui si userà φ per la sezione aurea e ϕ per il numero aureo.

```
\begin{displaymath}
   x_{1,2}= \frac{-b\pm\sqrt{b^2-4ac}}{2a}
\end{displaymath}
```

L'ambiente[2] *displaymath* dice "metti in display l'espressione matematica che racchiudo". Il segno `_` serve per dire "metti in posizione di deponente quanto contenuto nel prossimo gruppo"; il gruppo è delimitato dalle parentesi graffe, come si è già avuto modo di osservare; `\frac` sta per *fraction*; è una abbreviazione di una parola inglese, ma è una parola quasi completa; essa dà l'ordine di comporre una frazione; quindi bisogna specificare il numeratore e il denominatore mediante le rispettive espressioni racchiuse nei rispettivi gruppi; all'interno del numeratore compare il comando `\pm` che vuol dire "metti un segno 'più o meno' "; compare anche il comando `\sqrt`, sigla inglese che sta per *square root*, radice quadrata, ed esso comanda di mettere sotto radice il contenuto del gruppo che lo segue; il segno `^` vuol dire "metti ad esponente il contenuto del prossimo gruppo", ma in questo caso il gruppo non è evidenziato con le graffe, perché è costituito da un solo token.

Come si vede la scrittura di una breve espressione è semplicissima e non richiede una memoria particolare per ricordare i comandi da usare. Quando ci sono da scrivere delle lettere greche esse si scrivono con i loro nomi per disteso (in inglese) e le maiuscole si scrivono con l'iniziale maiuscola; per non confondere le sequenze di lettere che formano i nomi delle lettere greche come semplici variabili matematiche scritte in lettere latine, i nomi delle lettere greche sono preceduti dal segno '\': `\alpha`, `\beta`, `\gamma`, ..., `\Gamma`, `\Delta`, eccetera.

Per gli operatori matematici più comuni bastano i tasti della tastiera di un qualunque PC o laptop. Se il PC o il laptop è dotato di una tastiera estesa o avanzata, è possibile comporre da tastiera anche altri segni che non sono serigrafati sui tasti; bisogna solamente premere contemporaneamente uno o più tasti di controllo (Alt, Shift, Ctrl, ecc.) e almeno un tasto 'ordinario'. In ogni caso nelle prossime pagine appaiono un certo numero di tabelle che contengono la lista completa dei comandi matematici di LaTeX che si possono introdurre senza disporre di nessuna tastiera speciale.

Gli spazi che in matematica compaiono nel file sorgente non hanno nessuna rilevanza; LaTeX mette gli spazi matematici dove sono necessari e, anche se offre dei comandi per inserire esplicitamente alcuni tipi di spazi matematici, l'utente è vivamente invitato a non farne uso! LaTeX compone benissimo da solo, senza che l'utente debba perdere tempo per rendere il 99% delle volte il frutto della sua composizione meno bello e meno professionale di quello che LaTeX avrebbe fatto da solo.

Esistono alcuni pochi casi ben noti in cui l'utente può inserire degli spazi matematici; essi riguardano tutti gli operatori obliqui quando gli oggetti che li precedono o li seguono hanno forme particolari, oppure quando l'utente deve lavorare con integrali multipli e non vuole usare il pacchetto di estensione

[2] I comandi `\[` e `\]` sostituiscono efficacemente i comandi di apertura e di chiusura dell'ambiente *displaymath*.

amsmath, di cui si parlerà nel prossimo capitolo. Quindi il compositore sarà autorizzato a correggere gli spazi usati da LaTeX solo dopo aver corretto le bozze e se vuol aggiungere quei rarissimi tocchi di qualità che LaTeX non è riuscito a mettere da solo.

Per aiutare l'utente ad avere sottomano tutti i simboli che LaTeX mette a disposizione (e sono moltissimi) sono state preparate alcune tabelle di segni accostati ai comandi che li producono; si vedano allora le tabelle 13.1–13.8

Va detto subito che nelle tabelle 13.2, 13.3 e 13.8 i segni con uno sfondo grigio non sono usabili se non si usa il pacchetto di estensione *amsfonts* oppure il pacchetto di estensione *latexsym*; in realtà questo secondo pacchetto rende accessibili alcuni segni che con il vecchio LaTeX 2.09 erano direttamente accessibili senza estensioni; perciò questo pacchetto è da considerare come una maniera per compilare vecchi file sorgente predisposti per il vecchio LaTeX. Con il nuovo LaTeX 2_ε questi segni sono tutti accessibili tramite il pacchetto *amsfonts*. Siccome il pacchetto fa parte della collezione a cui appartiene anche *amsmath*, e questo è utilissimo per la composizione di testi che fanno uso di molta matematica, questi due pacchetti vengono solitamente caricati comunque, quindi non è un gran problema disporre di quei simboli. Tuttavia bisogna ancora segnalare che alcuni, come per esempio $\mho$, non dovrebbero mai venire usati perché contrari alle norme ISO, quindi non è una gran perdita se non se ne fa uso.

13.3 Alcune annotazioni sulle lettere greche

Per quanto riguarda le lettere greche, si può notare che alcune di esse hanno delle forme varianti i cui nomi cominciano con l'abbreviazione `var`. Esse sono delle varianti per quel che riguarda gli Stati Uniti, ma, tolto ϖ che non si usa mai, forse nemmeno negli Stati Uniti, gli altri segni sono quelli usati normalmente in Europa e in Italia; in alcuni file di classe che lo scrivente ha creato, egli ha scambiato i nomi varianti con quelli ordinari per rendere normali quelli che vengono usati di solito in Europa.

Inoltre in tutto il mondo è consuetudine scrivere le lettere greche maiuscole con caratteri diritti e non inclinati, nonostante le norme ISO prescrivano l'uso delle lettere inclinate per qualsiasi variabile matematica o fisica.

Se si desidera scrivere le lettere greche maiuscole con caratteri inclinati bisogna fare esattamente come è suggerito nella tabella 13.1. Oppure, se si sta facendo uso del pacchetto *amsmath*, si ottengono le lettere inclinate usando i comandi varianti (non visibili in nessuna delle tabelle citate) dove il nome delle lettere greche con iniziale maiuscola è preceduto da `var`. Scrivendo quindi `\varGamma`, `\varDelta`, `\varTheta`, eccetera, si ottengono i simboli $\varGamma$, $\varDelta$, $\varTheta$, ...

Minuscole					
\alpha	α	\iota	ι	\rho	ρ
\beta	β	\kappa	κ	\sigma	σ
\gamma	γ	\lambda	λ	\tau	τ
\delta	δ	\mu	μ	\upsilon	υ
\epsilon	ϵ	\nu	ν	\phi	ϕ
\zeta	ζ	\xi	ξ	\chi	χ
\eta	η	o	o	\psi	ψ
\theta	θ	\pi	π	\omega	ω
Varianti delle minuscole					
\varepsilon	ε	\varpi	ϖ	\varsigma	ς
\vartheta	ϑ	\varrho	ϱ	\varphi	φ
Maiuscole					
\Gamma	Γ	\Xi	Ξ	\Phi	Φ
\Delta	Δ	\Pi	Π	\Psi	Ψ
\Theta	Θ	\Sigma	Σ	\Omega	Ω
\Lambda	Λ	\Upsilon	Υ		
Maiuscole corsive					
\mathit{\Gamma}	$\mathit{\Gamma}$	\mathit{\Xi}	$\mathit{\Xi}$	\mathit{\Phi}	$\mathit{\Phi}$
\mathit{\Delta}	$\mathit{\Delta}$	\mathit{\Pi}	$\mathit{\Pi}$	\mathit{\Psi}	$\mathit{\Psi}$
\mathit{\Theta}	$\mathit{\Theta}$	\mathit{\Sigma}	$\mathit{\Sigma}$	\mathit{\Omega}	$\mathit{\Omega}$
\mathit{\Lambda}	$\mathit{\Lambda}$	\mathit{\Upsilon}	$\mathit{\Upsilon}$		

Tabella 13.1: Le lettere greche. La "omicron" di fatto è la lettera latina, che d'altra parte non si usa mai in matematica.

13.4 Alcune osservazioni sugli operatori funzionali

Gli operatori funzionali come sin, arctan, eccetera sono presentati nella tabella 13.4; essi *devono* essere usati esattamente con quei comandi per due importanti motivi.

1. I nomi che vengono scritti nel documento durante l'esecuzione di quei comandi sono quelli prescritti dalle norme internazionali ISO e, siccome essi sono indicati anche dalle norme nazionali UNI e queste in Italia hanno valore di legge, non è consentito usare nomi diversi, anche se si è abituati a scrivere 'tg' al posto di 'tan' oppure 'sen' al posto di 'sin'. In realtà le norme ISO prescrivono nomi che costituiscono abbreviazioni di parole o locuzioni latine, quindi *Ubi maior, minor cessat*!

2. Gli operatori, in quanto tali, richiedono certi spazi alla loro destra e alla loro sinistra; questi spazi dipendono dalla natura degli oggetti matematici che precedono o seguono gli operatori; solo usando quei comandi LaTeX sa che sta usando degli operatori e sa quali spazi usare.

\approx	≈	\asymp	≍	\bowtie	⋈
\cong	≅	\dashv	⊣	\doteq	≐
\downarrow	↓	\Downarrow	⇓	\equiv	≡
\frown	⌢	\ge	≥	\geq	≥
\gets	←	\gg	≫	\hookleftarrow	↩
\hookrightarrow	↪	\iff	⟺	\in	∈
\Join	⋈	\le	≤	\leadsto	⇝
\leftarrow	←	\Leftarrow	⇐	\leftharpoondown	↽
\leftharpoonup	↼	\leftrightarrow	↔	\Leftrightarrow	⇔
\leq	≤	\ll	≪	\longleftarrow	⟵
\Longleftarrow	⟸	\longleftrightarrow	⟷	\Longleftrightarrow	⟺
\longmapsto	⟼	\longrightarrow	⟶	\Longrightarrow	⟹
\mapsto	↦	\mid	∣	\models	⊨
\ne	≠	\nearrow	↗	\neq	≠
\ni	∋	\not=	≠	\nwarrow	↖
\parallel	∥	\perp	⊥	\prec	≺
\preceq	⪯	\propto	∝	\rightarrow	→
\Rightarrow	⇒	\rightharpoondown	⇁	\rightharpoonup	⇀
\rightleftharpoons	⇌	\searrow	↘	\sim	∼
\simeq	≃	\smile	⌣	\sqsubset	⊏
\sqsubseteq	⊑	\sqsupset	⊐	\sqsupseteq	⊒
\subset	⊂	\subseteq	⊆	\succ	≻
\succeq	⪰	\supset	⊃	\supseteq	⊇
\swarrow	↙	\to	→	\uparrow	↑
\Uparrow	⇑	\vdash	⊢		

Tabella 13.2: Gli operatori di relazione

\amalg	∐	\ast	∗	\bigcirc	◯
\bigtriangledown	▽	\bigtriangleup	△	\bullet	•
\cap	∩	\cdot	·	\circ	∘
\cup	∪	\dagger	†	\ddagger	‡
\diamond	⋄	\div	÷	\lhd	◁
\mp	∓	\odot	⊙	\ominus	⊖
\oplus	⊕	\oslash	⊘	\otimes	⊗
\pm	±	\rhd	▷	\setminus	\
\sqcap	⊓	\sqcup	⊔	\star	⋆
\times	×	\unlhd	⊴	\unrhd	⊵
\uplus	⊎	\vee	∨	\wedge	∧
\wr	≀				

Tabella 13.3: Gli operatori binari

La tabella 13.4 suddivide gli operatori in due categorie: (*a*) quelli chiamati 'operatori senza limiti' nei quali le indicazioni di esponente o di deponente

Operatori senza limiti				
\arccos	\arcsin	\arctan	\arg	\cos
\cosh	\cot	\coth	\csc	\deg
\dim	\exp	\hom	\ker	\lg
\ln	\log	\sec	\sin	\sinh
\tan	\tanh			
Operatori con limiti				
\det	\gcd	\inf	\lim	\liminf
\limsup	\max	\min	\Pr	\sup

Tabella 13.4: Gli operatori funzionali

\bigcap	$\bigcap$	$\cap$	\bigvee	$\bigvee$	$\vee$
\bigcup	$\bigcup$	$\cup$	\bigwedge	$\bigwedge$	$\wedge$
\bigodot	$\bigodot$	$\odot$	\coprod	$\coprod$	$\amalg$
\bigoplus	$\bigoplus$	$\oplus$	\int	$\int$	$\int$
\bigotimes	$\bigotimes$	$\otimes$	\oint	$\oint$	$\oint$
\bigsqcup	$\bigsqcup$	$\sqcup$	\prod	$\prod$	$\prod$
\biguplus	$\biguplus$	$\uplus$	\sum	$\sum$	$\sum$

\sqrt	$\sqrt{}$	$\sqrt{}$	$\sqrt{}$	$\sqrt{}$	$\sqrt{}$

Tabella 13.5: I grandi operatori

vengono usate normalmente come tali, e (*b*) quelli chiamati 'operatori con limiti' per i quali le indicazioni di esponente e di deponente vengono usati come nei simboli di sommatoria; si noti la differenza nell'equazione 13.1:

$$\lim_{x\to 0}\sum_{k=0}^{\infty}\sin^2 2kx = 0 \tag{13.1}$$

scritta facendo uso dei segni _ e ^ che normalmente indicano rispettivamente l'inserzione di un deponente o di un esponente:

```
\lim_{x\to 0}\sum_{k=0}^\infty \sin^2 2kx = 0
```

Si noti che *in modo testo* LaTeX compone i 'limiti' degli operatori con limiti, delle sommatorie, degli integrali, e simili, come se fossero dei deponenti o degli

((	/ /	\langle ⟨	\rangle ⟩
))	\lfloor ⌊	\rfloor ⌋	\lgroup ⟮
[[	\lceil ⌈	\rceil ⌉	\rgroup ⟯
]]	\backslash \	\vert |	\bracevert ⎪
\\| ‖	\Vert ‖	\updownarrow ↕	\Updownarrow ⇕
| |	\uparrow ↑	\Uparrow ⇑	\downarrow ↓
} }	\arrowvert |	\Arrowvert ‖	\Downarrow ⇓
{ {	\lmoustache ⎰	\rmoustache ⎱	

Tabella 13.6: I grandi delimitatori

esponenti; esso si comporta in questo modo per evitare di far diventare troppo grandi (alte e profonde) le righe con gli operatori con limiti che obbligherebbero a inserire di una generosa interlinea; in modo display, invece tutto procede come esemplificato nell'equazione 13.1.

Se si desidera definire nuovi operatori funzionali oltre a quelli che compaiono nella tabella 13.4, il pacchetto *amsmath* mette a disposizione due comandi con la seguente sintassi:

```
\DeclareMathOperator{⟨operatore⟩}{⟨definizione⟩}
\DeclareMathOperator*{⟨operatore⟩}{⟨definizione⟩}
```

dove ⟨*operatore*⟩ è il comando con il quale si vuole invocare l'operatore mentre la ⟨*definizione*⟩ contiene il modo di scrivere il nome dell'operatore. La dichiarazione

`\hat{x}`	$\hat{x}$	`\check{x}`	$\check{x}$
`\breve{x}`	$\breve{x}$	`\acute{x}`	$\acute{x}$
`\grave{x}`	$\grave{x}$	`\tilde{x}`	$\tilde{x}$
`\bar{x}`	$\bar{x}$	`\vec{x}`	$\vec{x}$
`\dot{x}`	$\dot{x}$	`\ddot{x}`	$\ddot{x}$
`\overline{x+y}`	$\overline{x+y}$	`\underline{x+y}`	$\underline{x+y}$
`\overbrace{x\cdots y}`	$\overbrace{x\cdots y}$	`\underbrace{x\cdots y}`	$\underbrace{x\cdots y}$
`\overrightarrow{x-y}`	$\overrightarrow{x-y}$	`\overleftarrow{x-y}`	$\overleftarrow{x-y}$
`\widehat{x-y}`	$\widehat{x-y}$	`\widetilde`	$\widetilde{x-y}$
`\mathring{x}`	$\mathring{x}$		

Tabella 13.7: Accenti e diacritici matematici

`\aleph`	$\aleph$	`\angle`	$\angle$	`\backslash`	$\backslash$
`\bot`	$\bot$	`\Box`	$\Box$	`\clubsuit`	$\clubsuit$
`\Diamond`	$\Diamond$	`\diamondsuit`	$\diamondsuit$	`\ell`	ℓ
`\emptyset`	$\emptyset$	`\exists`	$\exists$	`\flat`	$\flat$
`\forall`	$\forall$	`\hbar`	$\hbar$	`\heartsuit`	$\heartsuit$
`\Im`	$\Im$	`\imath`	$\imath$	`\infty`	∞
`\jmath`	$\jmath$	`\mho`	$\mho$	`\nabla`	∇
`\natural`	$\natural$	`\neg`	$\neg$	`\partial`	∂
`\prime`	$\prime$	`\Re`	$\Re$	`\sharp`	$\sharp$
`\spadesuit`	$\spadesuit$	`\surd`	$\surd$	`\top`	$\top$
`\triangle`	$\triangle$	`\wp`	$\wp$	`\\|`	$\|$

Tabella 13.8: Altri simboli

asteriscata definisce il comando per un operatore con limiti, mentre quella non asteriscata lo definisce per un operatore senza limiti.

Per esempio, l'operatore per separare la parte reale di un numero complesso è `\Re` e produce il simbolo $\Re$; volendolo ridefinire in modo che esso sia scritto in caratteri non gotici, ma sia in tondo, si potrebbe usare la dichiarazione[3]:

```
\DeclareMathOperator{\Re}{Re}
```

In questo modo lo stesso comando `\Re` non produce più $\Re z = \Re(x + \mathrm{i}\, y) = x$ ma $\operatorname{Re} z = \operatorname{Re}(x + \mathrm{i}\, y) = x$ e tratta il nuovo simbolo come un operatore senza limiti lasciandovi attorno gli stessi spazi che lascia normalmente attorno agli operatori. Si noti che anche l'unità immaginaria 'i' è un operatore (vettoriale) ed è stato

[3]Il comando `\Re` è già definito e i vari comandi `\Declare...` non producono messaggi d'errore se si ridefiniscono comandi esistenti; controllano però l'esistenza di un comando pre-esistente e, nel caso, scrivono un avvertimento nel file `.log`.

stampato facendo uso del comando `\uimm` la cui definizione un po' particolare verrà discussa nella pagina 332.

Si noti che non è necessario specificare il font di forma tonda, perché `\DeclareOperatorName` usa il font predefinito per gli operatori, ché è generalmente il tondo chiaro. Certo se uno desiderasse l'uso di un font diverso, fra quelli che si possono usare in matematica, lo deve specificare espressamente; se per esempio si volesse scrivere l'operatore `\Re` in tondo nero bisognerebbe dare la definizione:

```
\DeclareMathOperator{\Re}{\mathbf{Re}}
```

Di solito, però, non è una buona idea cambiare forma o serie rispetto alle forme o alle serie predefinite, perché queste in matematica hanno significati speciali ed è meglio evitare situazioni in cui si possano manifestare degli equivoci.

13.5 Alcune osservazioni sui grandi operatori

I grandi operatori hanno almeno due dimensioni, una 'normale' adatta al modo testo e una più grande adatta al modo display; la radice quadrata, invece ha numerose varianti adatte alla grandezza del radicando.

Così in modo testo si potrà scrivere con l'operatore piccolo (scelto automaticamente da LaTeX senza che il compositore se ne debba preoccupare) $\sum_{k=0}^{\infty} x^k = 1/(1-x)$ mentre in display sarà:

$$\sum_{k=0}^{\infty} x^k = \frac{1}{1-x} \tag{13.2}$$

Bisogna fare attenzione ad un dettaglio non trascurabile. Mentre la maggior parte delle volte LaTeX usa correttamente (almeno nel senso delle nostre aspettative) i grandi operatori scegliendo l'operatore grande oppure quello piccolo, talvolta ci sembra che si sbagli; si osservi l'equazione 13.3:

$$\frac{1}{\sum_{k=0}^{\infty} x^k} = 1 - x \tag{13.3}$$

Si nota che, nonostante l'equazione sia in display, LaTeX ha usato l'operatore del modo testo. A noi sembra scorretto, ma LaTeX ha agito correttamente, perché, nonostante si trovi a comporre una formula in display, se usasse la forma grande dell'operatore, l'intera formula risulterebbe squilibrata:

$$\frac{1}{\displaystyle\sum_{k=0}^{\infty} x^k} = 1 - x \tag{13.4}$$

come si vede bene nell'equazione 13.4.

LaTeX si comporta in questo modo tutte le volte che cerca di mantenere più o meno costante l'altezza delle righe nelle espressioni matematiche; esso infatti passa al modo testo anche per scrivere gli elementi delle matrici; bisogna ricordarsene e, se non ci piacesse, sarebbe nostra cura modificare il modo di scrivere le formule in modo da ottenere una forma grafica gradevole e nello stesso tempo corretta; si veda a questo proposito l'equazione 13.6.

13.6 I grandi delimitatori

Nella tabella 13.6 sono rappresentati i grandi delimitatori, o meglio i delimitatori estensibili; questi si possono estendere in altezza in modo virtualmente illimitato e possono racchiudere espressioni di qualunque grandezza; si pensi anche solo ai delimitatori che differenziano una matrice da un determinante. Ma i delimitatori possono essere resi estensibili solo premettendo al delimitatore di sinistra il comando `\left` e al delimitatore di destra il comando `\right`; questi due comandi devono essere sempre appaiati nella stessa espressione matematica, quindi se si deve mettere un solo delimitatore, l'altro lo si rende invisibile scrivendovi un 'punto' invece del segno di una parentesi; si veda l'equazione 13.5:

$$x_{1,2} = \begin{cases} \dfrac{-b \pm \sqrt{b^2 - 4ac}}{2a} & \text{se} \quad b^2 - 4ac > 0 \\ -\dfrac{b}{2a} & \text{se} \quad b^2 - 4ac = 0 \\ \dfrac{-b \pm \mathrm{i}\sqrt{4ac - b^2}}{2a} & \text{se} \quad b^2 - 4ac < 0 \end{cases} \tag{13.5}$$

composta scrivendo

```
x_{1,2} = \left\{ ... \right.
```

Per comporre la struttura dell'equazione 13.5, già più complessa di quelle usate come esempio fino ad ora, si può usare l'ambiente *cases* fornito dal pacchetto *amsmath*; con questo ambiente non è il caso di preoccuparsi di usare delimitatori invisibili perché ci pensa lui; anzi, questa osservazione ci fa capire perché è meglio servirsi di ambienti di composizione appositamente predisposti dal file di classe o dai file di estensione o da definizioni create dall'utente, piuttosto che ripetere ogni volta l'intero assemblaggio di comandi e di istruzioni, col rischio di dimenticarne qualcuno o di non essere coerenti ogni volta che si deve comporre lo stesso genere di struttura nella stessa maniera.

Ma `\left` e `\right` sono comodi anche per scrivere correttamente in modo display una formula come l'equazione 13.4 che diventa:

$$\left(\sum_{k=0}^{\infty} x^k \right)^{-1} = 1 - x \tag{13.6}$$

Bisogna prestare una particolare attenzione quando si deve spezzare una equazione fra più righe; quando si è in modo testo si suppone che le espressioni stampate siano molto semplici e non troppo lunghe; se l'espressione cade in fin di riga LaTeX (con molta riluttanza) provvede a dividere l'espressione dopo un operatore di relazione o dopo un operatore binario. Tuttavia sta al compositore evitare che si debba andare a capo mentre si è all'interno di una parentesi. Quando invece si è in display le espressioni possono essere anche molto lunghe e potrebbe essere necessario andare a capo all'interno di una parentesi. Solo chi ha scritto l'equazione ne conosce il ritmo di lettura e solo lui sa dove è meno grave andare a capo. Tuttavia resta la regola principale che si può andare a capo dopo gli operatori di relazione e, subordinatamente, dopo gli operatori binari, mai dopo gli operatori funzionali o i grandi operatori o immediatamente dopo i delimitatori di apertura. Ecco perché sono utili le tabelle 13.3–13.6, così anche chi non è matematico sa di che cosa si sta parlando.

Nella disgraziata ipotesi che fosse necessario andare a capo all'interno di una lunga espressione fra parentesi, allora potrebbero risultare utili i seguenti comandi, che, come `\left` e `\right` ingrandiscono i delimitatori, ma di una quantità fissa, indipendentemente da che cosa essi delimitino; non è difficile ottenere:

$$\Biggl(\biggl(\Bigl(\bigl(()\bigr)\Bigr)\biggr)\Biggr)$$

con i comandi

```
\Biggl(\biggl(\Bigl(\bigl(()\bigr)\Bigr)\biggr)\Biggr)
```

La lettera `l` o `r` alla fine di ogni comando specifica se il segno che segue è un delimitatore di sinistra (*left*) o di destra (*right*); esiste anche un suffisso `m` per i separatori che talvolta si trovano a metà di certe espressioni matematiche.

13.7 Accenti e segni diacritici matematici

Per segnare con 'diacritici' matematici i vari simboli che si usano, esistono i comandi riportati nella tabella 13.7; sarebbe un grave errore usare quegli accenti per inserire diacritici in modo testo. Per questi si veda la tabella 29.1. Per mettere accenti sulle lettere 'i' e 'j' in matematica bisogna ricorrere a due particolari versioni di questi segni privi del puntino: `\imath` e `\jmath`. Usando questi segni si può scrivere per esempio $\vec{a} = a_x\vec{\imath} + a_y\vec{\jmath}$ con il codice[4]:

```
\( \vec a = a_x \vec\imath + a_y \vec\jmath \)
```

Ma fra i segni che si possono mettere sopra o sotto singoli simboli o addirittura gruppi di simboli o espressioni matematiche, si possono notare in particolare

[4] I comandi `\imath` e `\jmath` per l'interprete sono ciascuno un solo token, anche se a noi umani sembrano sequenze di segni e di lettere e quindi ci appaiono ciascuno come una moltitudine di token.

`\overbrace` e `\underbrace`, che hanno proprietà diverse dagli altri diacritici: essi non servono solamente per mettere sopra o sotto una espressione matematica una graffa orizzontale, ma servono anche per applicare una annotazione all'espressione, una annotazione che di solito è costituita da un'altra breve espressione matematica; infatti la loro sintassi è rispettivamente:

```
\overbrace{⟨espressione matematica⟩}^{⟨annotazione⟩}
\underbrace{⟨espressione matematica⟩}_{⟨annotazione⟩}
```

e l'⟨*annotazione*⟩ (matematica) viene collocata rispettivamente sopra o sotto l'⟨*espressione principale*⟩ nel corpo degli indici di primo livello.

Va da sé che se l'annotazione è testuale l'⟨*annotazione*⟩ deve essere scritta in ambiente testo; quindi, se si sta usando il modulo *amsmath*, si può scrivere la nota nella forma `\text{`⟨*annotazione*⟩`}`; se invece non si sta usando *amsmath* bisogna ricorrere a una "scatola", come `\mbox`, nella quale si deve anche specificare a mano il corpo del font usato per la nota, per esempio `\mbox{\scriptsize` ⟨*annotazione*⟩`}`.

Sarebbe un errore usare il comando `\mathrm` per scrivere in tondo l'annotazione perché questa verrebbe scritta in tondo ma in modo matematico. Si confronti

- "valori efficaci", scritto con `\mbox{valori efficaci}`, con
- "valoriefficaci", scritto con `\mathrm{valori efficaci}`.

13.8 Gli ambienti matematici

Per inserire una espressione matematica in linea con il testo bisogna entrare in modo matematico; la maniera più semplice, ma sconsigliata, è quella di racchiudere i comandi matematici all'interno di un gruppo costituito da due segni di dollaro `$`. Così per scrivere che $\sin(\alpha+\beta) = \sin\alpha\cos\beta + \cos\alpha\sin\beta$, basta inserire nel testo sorgente

```
$\sin(\alpha+\beta) = \sin\alpha\cos\beta + \cos\alpha\sin\beta $
```

Ci si rende conto, però, che il segno di dollaro `$` funziona come un interruttore che cambia modo da testo a matematico o viceversa, senza preoccuparsi di controllare se l'operazione sia corretta; LaTeX offre il modo di eseguire questo controllo se si racchiude l'espressione matematica fra i comandi `\(` e `\)`, così:

```
\(
   \sin(\alpha+\beta) = \sin\alpha\cos\beta + \cos\alpha\sin\beta
\)
```

Non è necessario andare a capo dopo `\(` e prima di `\)`, ma il testo sorgente è più facilmente leggibile.

Il comando `\(` può essere dato solo mentre si è in modo testo e `\)` solo quando si è in modo matematico; se ciò non succedesse, LaTeX provvederebbe ad emettere gli opportuni messaggi d'errore che consentirebbero di eseguire le necessarie correzioni con facilità.

Analogamente in modo display sarebbe possibile usare i comandi di basso livello costituiti da due coppie di segni di dollaro; per scrivere:

$$\sin(\alpha + \beta) = \sin\alpha\cos\beta + \cos\alpha\sin\beta$$

basterebbe scrivere nel file sorgente:

```
$$
   \sin(\alpha+\beta) = \sin\alpha\cos\beta + \cos\alpha\sin\beta
$$
```

ma di nuovo mancherebbe la diagnostica di LaTeX. Va notato che aprire e chiudere una espressione matematica fra due coppie di dollari è incompatibile con altri pacchetti basati sulla sintassi di LaTeX; fra questi il pacchetto ***amsmath*** che è estremamente importante per la composizione della matematica avanzata – ma, direi, anche per la matematica semplice perché offre comandi e ambienti utilissimi e non disponibili nel nucleo di LaTeX. La delimitazione delle espressioni matematiche in display mediante le coppie di dollari è anche incompatibile con le opzioni e i pacchetti che consentono di comporre le espressioni accostate a sinistra.

LaTeX, invece, offre due ambienti per scrivere espressioni senza la numerazione e un ambiente per scrivere espressioni numerate:

```
\[
   \sin(\alpha+\beta) = \sin\alpha\cos\beta + \cos\alpha\sin\beta
\]
```

con la sintassi:

`\[` ⟨*espressione matematica*⟩`\]`

oppure con l'ambiente *displaymath*:

```
\begin{displaynmath}
   \sin(\alpha+\beta) = \sin\alpha\cos\beta + \cos\alpha\sin\beta
\end{displaymath}
```

che segue la sintassi:

`\begin{displaymath}`
⟨*espressione matematica*⟩
`\end{displaymath}`

Per le espressioni numerate si usa l'ambiente *equation* con la sintassi:

```
\begin{equation}
⟨espressione matematica⟩
\label{⟨etichetta⟩}
\end{equation}
```

Usando il pacchetto `amsmath` diventa disponibile l'ambiente *equation**:

```
\begin{equation*}
⟨espressione matematica⟩
\end{equation*}
```

del tutto equivalente all'ambiente *displaymath* solo che gestisce meglio gli spazi e accetta anche espressioni matematiche più complesse disponibili con quel pacchetto.

La sintassi è auto esplicativa, ma si noti che l'⟨*etichetta*⟩ può contenere un nome simbolico con il quale richiamare l'equazione per nome, invece che per numero, rendendo così automatico il processo dei riferimenti incrociati mediante i comandi `\label`, `\ref` e `\pageref`. Il pacchetto `amsmath` per citare le equazioni mette a disposizione anche il comando `\eqref` che provvede da solo a racchiuderne il riferimento vero fra parentesi tonde come si fa abitualmente (e come *non* si è fatto in questo testo). Quel pacchetto mette a disposizione anche un comando `\tag` per identificare le equazioni; esso consente di assegnare all'equazione un 'nome' letterale, invece che un identificativo numerico, ed è usabile anche all'interno dell'ambiente *equation**, benché questo si riferisca ad un ambiente non numerato; è proprio quello che si è fatto per l'equazione che definisce la sezione aurea nella pagina 304.

13.9 Le unità di misura

Quanto esposto nei paragrafi precedenti dovrebbe consentire di scrivere equazioni di una riga, numerate o non numerate e di poterle citare in modo corretto senza scambi di identità.

Prima di passare al capitolo successivo vale la pena di soffermarci su una questione molto importante, visto che la matematica non serve solo ai matematici puri, ma anche ai tecnologi, ai fisici, ai chimici, e a tanti altri tecnici e scienziati che scrivono equazioni fra grandezze misurabili con adeguati strumenti.

Le scienze sperimentali usano espressioni che coinvolgono grandezze misurabili e con queste la composizione della matematica deve soddisfare a ulteriori regole specificate dalle norme ISO–UNI.

1. Le grandezze vanno scritte in corsivo matematico; anche le costanti fisiche vanno scritte in corsivo matematico, a differenza delle costanti matematiche che vanno scritte in tondo;

2. quando si esplicita il valore numerico delle costanti fisiche o di qualunque grandezza, bisogna inserire anche le unità di misura.

3. Le unità di misura devono essere composte seguendo scrupolosamente le norme ISO, sia per i loro nomi, sia per i loro simboli e i loro prefissi decimali; inoltre

   (a) vanno sempre scritte dopo (a destra) del valore numerico che indica la misura della grandezza;

   (b) devono essere scritte in tondo anche nelle espressioni matematiche;

   (c) devono essere separate da uno spazio fine dal valore numerico;

   (d) questo spazio deve essere 'non separabile' nel senso che non si può andre 'a capo' fra la misura e l'unità di misura.

   Tutte queste azioni si ottengono in un colpo solo se si usa il comando `\unit`[5] disponibile con il pacchetto *babel* quando si scrive in italiano. Bisogna solo scrivere il comando `\unit` con il suo argomento 'attaccato' alla misura, per esempio `3,9\unit{k\ohm}` che produce 3,9 kΩ.[6]

4. I pedici e gli apici numerici vanno scritti con carattere diritto, ma quelli letterali vanno scritti in corsivo matematico se rappresentano quantità variabili, o in tondo se rappresentano delle apposizioni della grandezza: quindi $V_{\max}$ e non V_{max}; invece bisogna scrivere V_T se T è, per esempio, una temperatura; V_i se i rappresenta l'indice dell'i-esimo elemento di un insieme numerabile, ma V_{i} se 'i' sta per 'iniziale'. Per questo scopo *babel* in italiano offre il comando `\ped` per scrivere pedici in tondo; e `\ap` per scrivere apici in tondo; entrambi i comandi funzionano anche in modo testo per inserire pedici o apici, ma in modo testo mantengono il font in uso, senza preoccuparsi se sia diritto o inclinato.

Una tipica equazione fisica potrebbe essere:

$$i(t) = \sqrt{2} I_{\mathrm{eff}}\, \mathbf{Re}\left[\mathrm{e}^{\mathrm{i}\,\omega t}\right]$$

dove 'eff' sta per 'efficace', 'e' è la costante matematica base dei logaritmi naturali da distinguere dalla 'carica elettrica elementare e'; inoltre 'i' è l'unità immaginaria da distinguere dalla grandezza fisica 'corrente elettrica i'.

La figura 13.1 riprende due strafalcioni giornalistici pubblicati su un quotidiano nazionale; nella figura di sinistra c'è l'errore frequentissimo di scrivere 'K',

[5] Il comando `\unit` non è disponibile di default, ma, se si scrive in italiano, deve eventualmente essere reso disponibile dando nel preambolo il comando `\setISOcompliance`. Per contro, il modulo *italian* di *babel* si disinteressa di queste impostazioni se nel preambolo sono stati caricati i pacchetti che mettono a disposizioni dell'utente altri comandi per gestire correttamente le unità di misura in conformità con le norme ISO, per esempio il file di estensione *siunitx*.

[6] In modo matematico bisognerebbe scrivere `\Omega`, mentre in modo testo bisognerebbe scrivere `\textohm`. Non è difficile definire un comando `\ohm` che distingua che cosa si deve usare a seconda se si è in modo matematico o in modo testo, come è stato fatto in questo documento nella pagina 483.

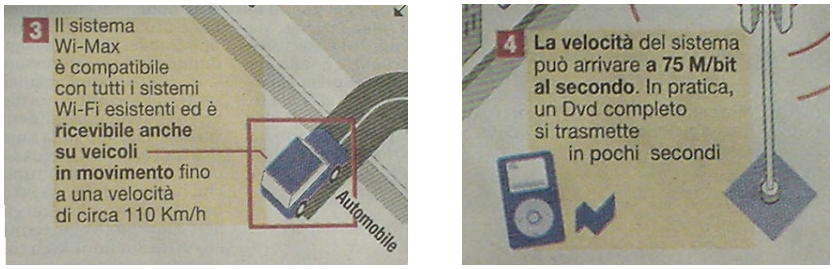


Figura 13.1: Due esempi di strafalcioni giornalistici

invece di 'k' per il prefisso 'kilo'; in effetti in quella figura si parla di kelvin metri all'ora... Nella figura a destra 'M/bit al secondo' doveva essere[7] M bit/s o forse Mi bit/s. Ammesso che si tratti solo di un errore di stampa, con una velocità di trasmissione di 75 Mi bit/s per trasmettere il contenuto di un DVD completo (che generalmente contiene più di 4,5 Gi byte, cioè più di 36 Gi bit) occorrono almeno 480 s cioè almeno 8 min, tutt'altro che pochi secondi...

Chi compone i suoi testi con LaTeX è tenuto ad astenersi da una simile ignoranza e sciatteria!

Egli, però, può trarre giovamento dall'uso dei pacchetti `units`, `SIunits` e `siunitx`. Il primo pacchetto definisce una versione del comando `\unit` (diversa da quella che è definita con l'opzione *italian* del pacchetto `babel`) che consente di specificare alcune opzioni di spaziatura fra la misura e l'unità di misura e consente di scrivere le frazioni mediante numeratore rialzato e denominatore ribassato, utili sia in linea con il testo, sia in matematica, affinché risulti ben chiaro chi sia il numeratore e chi il denominatore; si può quindi specificare una velocità nella forma $70\,^{\mathrm{km}}/_{\mathrm{h}}$; questo modo di scrivere è una pratica diffusa nei paesi dell'Europa del Nord, ma non è previsto dalle norme ISO.

Il secondo pacchetto contiene una bella e aggiornata descrizione della situazione relativa al Sistema Internazionale (SI) di unità di misura e richiama anche le norme tipografiche specificate dalla Conferenza Generale dei Pesi e delle Misure e dalle sue commissioni, nonché dalle norme ISO che regolano questa questione. Il pacchetto fornisce anche una diversa definizione del comando `\unit` che richiede però due argomenti e quindi è incompatibile con la definizione dell'opzione *italian*

[7] I prefissi stabiliti dalle norme ISO dal kilo in poi rappresentano potenze di 1000. In informatica è più conveniente fare riferimento alle potenze di 2; per indicare i multipli normalmente si usano gli stessi prefissi ISO (con k mutato in K) seguiti da una 'i'; essi rappresentano le potenze di $2^{10} = 1024$. Il prefisso 'Mi' indica quindi $(2^{10})^2$ e il prefisso 'Gi' indica il valore $(2^{10})^3$. Inoltre il simbolo né del bit né del byte sono codificati come unità di misura del SI; sarebbe opportuno scrivere queste grandezze per disteso, piuttosto che con 'b' e 'B', anche se gli informatici spesso usano queste abbreviazioni, che si prestano a essere fraintese; per esempio le compagnie telefoniche che concedono in uso le loro linee per collegamenti ADSL, le reclamizzano spesso con velocità strepitose, per esempio di 200 MB, o 200 mb, in entrambi i casi dimenticando di rispettare le norme; in realtà quelle velocità indicano il valore massimo di trasmissione di 200 Mi bit/s lasciando credere ai clienti che si tratti di 200 Mi byte/s, quando invece sono otto volte più basse.

di *babel*; all'interno del secondo argomento è possibile specificare le unità di misura a parole (in inglese britannico e al singolare) per cui è possibile indicare la velocità di 70 km/h con la scrittura:

```
\unit{70}{\kilo\metre\per\hour}
```

Invece con il comando definito con l'opzione *italian* di *babel* basta scrivere `70\unit{km/h}`.

Usando il pacchetto *SIunits* è possibile specificargli l'opzione *italian* cosicché si evita la collisione con la definizione di `\unit` in *babel*, perché la definizione del pacchetto *SIunits* diventa `\unita` (in italiano e senza accento) e i due comandi con le rispettive sintassi possono essere usati contemporaneamente. Viceversa, se questa soluzione non interessa, basta caricare il pacchetto *babel* *dopo* il pacchetto *SIunits* perché *babel* è predisposto affinché non ridefinisca il comando `\unit` di *SIunits*.

Lo stesso autore di *SIunits*, D.N.J. Els, ha preparato un secondo pacchetto, *SIstyle*, per comporre correttamente le indicazioni metriche delle grandezze fisiche; secondo questo pacchetto il comando per comporre la grandezza fisica è semplicemente:

`\SI{`⟨*misura*⟩`}{`⟨*unità di misura*⟩`}`

Il pregio del pacchetto, oltre alla semplicità d'uso, che fa dire ad Els di preferire egli stesso questo pacchetto al suo più esteso pacchetto *SIunits*, consiste nelle vaste possibilità di personalizzazione e di adeguamento a specifiche norme nazionali; serve, cioè, ad impostare lo stile di scrittura delle grandezze in modo da assicurare l'uniformità di composizione all'interno del documento.

Va infine segnalato anche il recentissimo quarto pacchetto *siunitx* che sostituisce tutti i precedenti e che offre molte possibilità in più per gestire le grandezze fisiche in termini numerici; dispone di un modulo di elaborazione dei numeri che consente anche scritture del tipo `30e3` e che le macro di scrittura trasformano in 30×10^3; questo permette di trarre i valori numerici da file scritti dagli stessi strumenti di misura moderni, dove i valori numerici sono espressi con la notazione informatica dei numeri a virgola mobile. Si può specificare il numero di cifre da scrivere nei valori numerici delle misure: il modulo di elaborazione dei numeri provvede ad arrotondare quel valore numerico al numero richiesto di decimali. Infatti se si specifica che si vogliono consistentemente 4 decimali e la virgola decimale, il numero `1.234567` viene stampato nella forma 1,2346 dove il numero di decimali è quello voluto ma l'ultima cifra tiene conto dell'arrotondamento in alto, visto che la parte scartata è maggiore della metà dell'unità corrispondente all'ultima cifra scritta; inoltre il punto decimale è consistentemente cambiato nella virgola, come richiesto. Si rinvia il lettore alla documentazione in `.../doc/latex/siunitx/siunitx.pdf`.

Il compositore stia bene attento alla preposizione 'per', che in italiano significa 'moltiplicato' e in inglese significa 'diviso'; una resistività di 'sette microohm per metro' si scrive $7\,\mu\Omega\,\mathrm{m}$ oppure $7\,\mu\Omega \cdot \mathrm{m}$, ma sarebbe sbagliatissimo scriverla $7\,\mu\Omega/\mathrm{m}$ interpretando la preposizione 'per' all'inglese.

Capitolo 14

LaTeX: la matematica avanzata

Questo capitolo è dedicato a chi ha bisogno di scrivere espressioni matematiche un po' più complesse di quelle che si possono comporre con i comandi base di LaTeX esposti nel capitolo precedente.

È comodo fare uso del pacchetto di estensione *amsmath*. Questa estensione è il frutto della collaborazione con l'American Mathematical Society (AMS) che, dai tempi del Plain TeX originale, anzi del TeX 78, la primissima versione del sistema TeX, ha sempre collaborato ed è sempre stata strenua sostenitrice di questo software, ne ha sponsorizzato il 'trade mark', ed è stata la prima società scientifica ad accettare i contributi alle proprie riviste scritti in Plain TeX, in $\mathcal{AMS}$ TeX e infine in $\mathcal{AMS}$ LaTeX; vale a dire con i programmi standard estesi con i pacchetti preparati dai programmatori della Società stessa e poi messi a disposizione di tutta la comunità degli utenti. L'American Mathematical Society ha anche messo a disposizione i file di classe specializzati per scrivere articoli destinati alle sue varie pubblicazioni; ora questa pratica è seguita da tutte le più importanti riviste scientifiche internazionali; anche gli atti di molti congressi sono scritti usando direttamente i file sorgente degli autori delle memorie presentate a quei congressi.

Il pacchetto principe per l'utente generico è il pacchetto *amsmath*. Questo pacchetto mette a disposizione una miriade di comandi nuovi, da quelli che servono per usare altri simboli, oltre a quelli già molto numerosi presenti normalmente in LaTeX, a quelli che servono per predisporre incolonnamenti particolari, a quelli che servono per comporre segni o strutture matematiche avanzate.

14.1 I simboli di *amsmath*

Il pacchetto *amsmath* consente di accedere ai simboli delle tabelle 14.1 e 14.2.

\ulcorner	⌜	\urcorner	⌝	\llcorner	⌞
\lrcorner	⌟	\dashrightarrow	⇢	\dashleftarrow	⇠
\widehat	^	\widetilde	~	\dasharrow	⇢
\rightleftharpoons	⇌	\leftrightharpoons	⇋	\angle	∠
\hbar	ℏ	\sqsubset	⊏	\sqsupset	⊐
\mho	℧	\square	□	\lozenge	◊
\vartriangleright	▷	\vartriangleleft	◁	\trianglerighteq	⊵
\triangleleftteq	⊴	\circleddash	⊝	\boxdot	⊡
\boxplus	⊞	\boxtimes	⊠	\boxminus	⊟
\blacksquare	■	\centerdot	·	\blacklozenge	⧫
\circlearrowright	↻	\circlearrowleft	↺	\Vdash	⊩
\Vvdash	⊪	\vDash	⊨	\twoheadrightarrow	↠
\twoheadleftarrow	↞	\leftleftarrows	⇇	\rightrightarrows	⇉
\upuparrows	⇈	\downdownarrows	⇊	\upharpoonright	↾
\downharpoonright	⇂	\upharpoonleft	↿	\downharpoonleft	⇃
\rightarrowtail	↣	\leftarrowtail	↢	\leftrightarrows	⇆
\rightleftarrows	⇄	\Lsh	↰	\Rsh	↱
\rightsquigarrow	⇝	\leftrightsquigarrow	↭	\looparrowleft	↫
\looparrowright	↬	\circeq	≗	\succsim	≿
\gtrsim	≳	\gtrapprox	⪆	\multimap	⊸
\therefore	∴	\because	∵	\doteqdot	≑
\triangleq	≜	\precsim	≾	\lesssim	≲
\lessapprox	⪅	\eqslantless	⪕	\eqslantgtr	⪖
\curlyeqprec	⋞	\curlyeqsucc	⋟	\preccurlyeq	≼
\leqq	≦	\leqslant	⩽	\lessgtr	≶
\backprime	‵	\risingdotseq	≓	\fallingdotseq	≒
\succcurlyeq	≽	\geqq	≧	\geqslant	⩾
\gtrless	≷	\vartriangleright	▷	\vartriangleleft	◁
\trianglerighteq	⊵	\triangleleftteq	⊴	\bigstar	★
\between	≬	\blacktriangledown	▼	\blacktriangleright	▶
\blacktriangleleft	◀	\vartriangle	△	\blacktriangle	▲
\triangledown	▽	\eqcirc	≖	\lesseqgtr	⋚
\gtreqless	⋛	\lesseqqgtr	⪋	\gtreqqless	⪌
\Rrightarrow	⇛	\Lleftarrow	⇚	\veebar	⊻
\barwedge	⊼	\doublebarwedge	⩞	\measuredangle	∡
\sphericalangle	∢	\varpropto	∝	\smallsmile	⌣
\smallfrown	⌢	\Subset	⋐	\Supset	⋑
\Cup	⋓	\Cap	⋒	\curlywedge	⋏
\curlyvee	⋎	\leftthreetimes	⋋	\rightthreetimes	⋌
\subseteqq	⫅	\supseteqq	⫆	\bumpeq	≏
\Bumpeq	≎	\lll	⋘	\ggg	⋙
\circledS	Ⓢ	\pitchfork	⋔	\dotplus	∔
\backsim	∽	\backsimeq	⋍	\complement	∁
\intercal	⊺	\circledcirc	⊚	\circledast	⊛

Tabella 14.1: Prima serie di simboli accessibili con il pacchetto *amsmath*

\lvertneqq	$\lvertneqq$	\gvertneqq	$\gvertneqq$	\nleq	$\nleq$
\ngeq	$\ngeq$	\nless	$\nless$	\ngtr	$\ngtr$
\nprec	$\nprec$	\nsucc	$\nsucc$	\lneqq	$\lneqq$
\gneqq	$\gneqq$	\nleqslant	$\nleqslant$	\ngeqslant	$\ngeqslant$
\lneq	$\lneq$	\gneq	$\gneq$	\npreceq	$\npreceq$
\nsucceq	$\nsucceq$	\precnsim	$\precnsim$	\succnsim	$\succnsim$
\lnsim	$\lnsim$	\gnsim	$\gnsim$	\nleqq	$\nleqq$
\ngeqq	$\ngeqq$	\precneqq	$\precneqq$	\succneqq	$\succneqq$
\precnapprox	$\precnapprox$	\succnapprox	$\succnapprox$	\lnapprox	$\lnapprox$
\gnapprox	$\gnapprox$	\nsim	$\nsim$	\ncong	$\ncong$
\diagup	$\diagup$	\diagdown	$\diagdown$	\varsubsetneq	$\varsubsetneq$
\varsupsetneq	$\varsupsetneq$	\nsubseteqq	$\nsubseteqq$	\nsupseteqq	$\nsupseteqq$
\subsetneqq	$\subsetneqq$	\supsetneqq	$\supsetneqq$	\varsubsetneqq	$\varsubsetneqq$
\varsupsetneqq	$\varsupsetneqq$	\subsetneq	$\subsetneq$	\supsetneq	$\supsetneq$
\nsubseteq	$\nsubseteq$	\nsupseteq	$\nsupseteq$	\nparallel	$\nparallel$
\nmid	$\nmid$	\nshortmid	$\nshortmid$	\nshortparallel	$\nshortparallel$
\nvdash	$\nvdash$	\nVdash	$\nVdash$	\nvDash	$\nvDash$
\nVDash	$\nVDash$	\ntrianglerighteq	$\ntrianglerighteq$	\ntrianglelefteq	$\ntrianglelefteq$
\ntriangleleft	$\ntriangleleft$	\ntriangleright	$\ntriangleright$	\nleftarrow	$\nleftarrow$
\nrightarrow	$\nrightarrow$	\nLeftarrow	$\nLeftarrow$	\nRightarrow	$\nRightarrow$
\nLeftrightarrow	$\nLeftrightarrow$	\nleftrightarrow	$\nleftrightarrow$	\divideontimes	$\divideontimes$
\varnothing	$\varnothing$	\nexists	$\nexists$	\Finv	$\Finv$
\Game	$\Game$	\eth	$\eth$	\eqsim	$\eqsim$
\beth	$\beth$	\gimel	$\gimel$	\daleth	$\daleth$
\lessdot	$\lessdot$	\gtrdot	$\gtrdot$	\ltimes	$\ltimes$
\rtimes	$\rtimes$	\shortmid	$\shortmid$	\shortparallel	$\shortparallel$
\smallsetminus	$\smallsetminus$	\thicksim	$\thicksim$	\thickapprox	$\thickapprox$
\approxeq	$\approxeq$	\succapprox	$\succapprox$	\precapprox	$\precapprox$
\curvearrowleft	$\curvearrowleft$	\curvearrowright	$\curvearrowright$	\digamma	$\digamma$
\varkappa	$\varkappa$	\Bbbk	$\Bbbk$	\hslash	$\hslash$
\backepsilon	$\backepsilon$				

Tabella 14.2: Seconda serie di simboli accessibili con il pacchetto *amsmath*

Se insieme a *amsmath* si invoca anche *amssymb* si hanno disponibili tra gli altri anche i vecchi segni di LaTeX 2.09, forniti anche dal pacchetto *ltxsymb* il cui uso, come si è avuto occasione di commentare, è oggi sconsigliato. D'altra parte *amssymb* invoca a sua volta *amsfonts* che mette a disposizione anche altri font, in particolare i font 'Euler fraktur' che in matematica possono talvolta risultare più efficaci dei caratteri calligrafici.

Invece, se non occorre tutto il macchinario di *amsmath* ma occorrono solo alcuni simboli, in particolare le lettere maiuscole ad aste raddoppiate, si può usare solo il pacchetto *amsfonts* che definisce il comando `\mathbb` per usarle; tuttavia si consiglia di caricare solo il pacchetto *amssymb* che a sua volta carica *amsfonts*. Qui, nelle tabelle citate, si è riportato tutto l'insieme dei simboli componibili per mezzo di tutti i pacchetti associati a *amsmath* in modo che

il lettore abbia la consapevolezza della potenza del sistema TeX per quanto riguarda la simbologia disponibile.

14.2 Le estensioni dei font matematici

La AMS ha messo a disposizione due altre polizze di font chiamate cripticamente `msam` e `msbm` i cui simboli sono direttamente accessibili con il pacchetto `amssymb`.

Nella tabella 14.2 si notano diversi operatori binari 'negati'; i font normali di LaTeX contengono anche il qualificatore `\not` che serve per creare per sovrapposizione[1] la negazione di qualsiasi operatore binario.

In linea di principio sarebbe desiderabile usare sempre i segni negati della tabella 14.2; talvolta però bisogna fare di necessità virtù e ricorrere a `\not`; per esempio non c'è preconfezionato il simbolo di 'non appartenenza', `\notin`; questo si può ottenere negando l'operatore `\in`: $a \notin b$, e il risultato appare pienamente accettabile. In altri casi, come per esempio per l'operatore 'non esiste', si può negare l'operatore 'esiste',`\exists`, ma se si fa così si ottiene: $\not\exists x$. Se invece si usa direttamente l'operatore `\nexists` il risultato è molto migliore: $\nexists x$, anzi, è perfetto.

Un'altra cosa a cui bisogna fare attenzione sono gli spazi che precedono e seguono gli operatori di vario genere a seconda di ciò che li precede o li segue; il discorso può risultare abbastanza complesso, ma è completamente descritto nel TeXbook, che riporta anche la tabella completa per tutte le combinazioni di accostamento dei vari operatori e dei vari simboli matematici. Generalmente lo spazio viene azzerato quando due operatori dello stesso tipo si susseguono; per esempio, per dire che l'ordinamento implicato dall'operatore $\leq$ viene esteso dall'ordinamento implicato da $\preceq$, non è corretto scrivere `\( \leq \subseteq \preceq \)`, ottenendo $\leq\subseteq\preceq$, perché gli spazi sono sbagliati; bisogna invece scrivere `\( {\leq} \subseteq {\preceq} \)`, perché le parentesi di gruppo nascondono all'operatore intermedio la natura del loro contenuto; infatti si ottiene: $\leq \subseteq \preceq$.

Questi sono punti delicati, forse troppo tecnici per la natura di questo testo, ma ricordiamoci: la tipografia è un'arte e la composizione tipografica della matematica ne costituisce un livello superiore. LaTeX può fare di tutto da solo, ma qualche volta ha bisogno di un piccolo aiuto da parte dell'utente.

14.3 I sistemi di equazioni

I sistemi di equazioni sono formati da un certo numero di equazioni incolonnate una sopra l'altra, di solito allineate in modo che i segni di uguaglianza (o qualunque altro segno di relazione) siano incolonnati verticalmente.

[1] Si veda il capitolo 22 dove si spiega che il comando `\not` va modificato in modo da renderlo compatibile con il formato archiviabile; un altro segno che va modificato è `\mapsto`.

LaTeX è nato con un ambiente *eqnarray* (anche asteriscato) che consente di eseguire questo incolonnamento; per esempio:

$$\begin{aligned} 3x + 7y - 2z &= 13 && (14.1)\\ 2y + 3z &= 12 && (14.2)\\ 5x - 3y + 4z &= 6 && (14.3) \end{aligned}$$

in cui le equazioni sono numerate da 14.1 a 14.3 e vi si può fare riferimento singolarmente usando il comando `\label` alla fine di ciascuna riga. Il sistema esposto è stato scritto con il codice

```
\begin{eqnarray}
  3x + 7y - 2z &=& 13 \label{equ:sistema1}\\
       2y + 3z &=& 12 \label{equ:sistema2}\\
  5x - 3y + 4z &=& 6  \label{equ:sistema3}
\end{eqnarray}
```

Se invece di usare l'ambiente *eqnarray* si fosse usato l'ambiente *eqnarray** le equazioni non sarebbero state numerate.

Purtroppo questo ambiente è nato con un 'peccato originale'; se il lettore osserva bene, constata che lo spazio a destra e a sinistra dei segni di uguaglianza nelle equazioni 14.1–14.3 è maggiore che non nelle equazioni del capitolo precedente. Non si sa se questo fosse voluto o se sia successo per caso, ma quando dal vecchio LaTeX 2.09 si è passati al nuovo LaTeX 2_ε, questo difetto è stato conservato, apparentemente per mantenere una compatibilità con i file predisposti per essere composti con la vecchia versione.

Sebbene non sarebbe difficile correggere la definizione del comando di apertura dell'ambiente *eqnarray* per disporre della spaziatura giusta ricorrendo alle funzionalità del pacchetto *xpatch*, oggi gli ambienti *eqnarray* e *eqnarray** non dovrebbero venire più usati, ma si dovrebbero usare sempre solo gli ambienti predisposti dal pacchetto *amsmath*.

14.4 Gli ambienti di composizione di *amsmath*

Il pacchetto *amsmath* mette a disposizione una serie di ambienti per comporre e/o incolonnare equazioni o per raccoglierle in modo ordinato, ovvero per scriverle su più righe in modo professionale, anche se l'intervento dell'autore per conoscere il ritmo delle espressioni matematiche è fondamentale.

Gli ambienti disponibili sono raggruppati nella tabella 14.3.

Tutti gli ambienti hanno la versione asteriscata che differisce da quella non asteriscata per la mancanza delle numerazione delle equazioni; solo *split* e *aligned* ne sono privi perché devono essere usati all'interno degli altri ambienti, non possono essere usati da soli. *subequations* invece ne è privo perché serve per numerare le equazioni in un modo particolare.

equation	*equation**	*align*	*align**
gather	*gather**	*flalign*	*flalign**
multline	*multline**	*alignat*	*alignat**
aligned	*split*	*subequations*	

Tabella 14.3: Gli ambienti di allineamento di `amsmath`

La numerazione può anche essere esclusa inserendo il comando `\notag` prima del segno di 'a capo' di ogni riga dell'allineamento; alternativamente il comando `\tag` consente di inserire una etichetta a propria scelta come per esempio nell'equazione:

$$a^2 + b^2 = c^2 \qquad \text{(Teorema di Pitagora)}$$

ottenuto scrivendo:

```
\begin{equation}
a^2 +b^2 = c^2 \tag{Teorema di Pitagora}
\end{equation}
```

Tutti gli ambienti di incolonnamento usano il segno `&` per passare da una colonna all'altra e non vogliono mai due segni a cavallo degli operatori di relazione, come succede con *eqnarray*, ma ne vogliono uno solo *prima* dell'operatore.

14.4.1 L'ambiente *equation*

Di *equation* si è già detto quanto basta; il suo compagno asteriscato, a parte la numerazione, consente di contenere tutti gli altri ambienti che possono essere inclusi negli ambienti numerabili, in particolare *split* per scrivere lunghe equazioni su più righe. La presenza di *equation** consente di evitare di usare *displaymath*; quest'ultimo infatti non garantisce le spaziature corrette e omogenee con quelle degli altri ambienti matematici. La definizione mediante l'asterisco è anche più coerente con l'uso dell'asterisco negli altri ambienti numerabili, ma di cui si desidera eliminare la numerazione.

14.4.2 L'ambiente *aligned*

L'ambiente *aligned* serve per comporre un allineamento di equazioni o di espressioni matematiche all'interno di un altro ambiente che sia in grado, eventualmente, di assegnare un numero a questo allineamento. Per esempio si può scrivere:

$$\begin{aligned}\mathrm{Ker}_n(x) = &-[\log(x/2)+\gamma] + \tfrac{1}{4}\pi\,\mathrm{Bei}_n(x)\\ &+\frac{1}{2}\sum_{k=0}^{n-1}\frac{(n-k-1)!(x/2)^{2k-n}}{k!}\cos\frac{(3n+2k)\pi}{4}\\ &+\frac{1}{2}\sum_{k=0}^{\infty}\frac{(x/2)^{n+2k}}{k!(n+k)!}[\Phi(k)+\Phi(n+k)]\cos\frac{(3n+2k)\pi}{4}\end{aligned} \tag{14.5}$$

Questo ambiente produce risultati molto simili a quelli dell'ambiente *split* a cui si rinvia per i commenti.

14.4.3 L'ambiente *split*

L'ambiente *split* serve per dividere una lunga espressione matematica fra più righe.

L'equazione su più righe 14.5, resa con *aligned*, con *split* diventa:

$$\begin{aligned}\mathrm{Ker}_n(x) = &-[\log(x/2)+\gamma] + \tfrac{1}{4}\pi\,\mathrm{Bei}_n(x)\\ &+\frac{1}{2}\sum_{k=0}^{n-1}\frac{(n-k-1)!(x/2)^{2k-n}}{k!}\cos\frac{(3n+2k)\pi}{4}\\ &+\frac{1}{2}\sum_{k=0}^{\infty}\frac{(x/2)^{n+2k}}{k!(n+k)!}[\Phi(k)+\Phi(n+k)]\cos\frac{(3n+2k)\pi}{4}\end{aligned} \tag{14.6}$$

Il risultato è stato ottenuto con

```
\begin{equation}
  \begin{split}
    \Ker_n(x) & = -[\log(x/2)+\gamma]
                          +{\textstyle\frac{1}{4}}\pi\Bei_n(x)\\
              & \quad+\frac{1}{2}\sum_{k=0}^{n-1}
                        \frac{(n-k-1)!(x/2)^{2k-n}}{k!}
                                        \cos\frac{(3n+2k)\pi}{4}\\
              & \qquad+\frac{1}{2}\sum_{k=0}^\infty
                \frac{(x/2)^{n+2k}}{k!(n+k)!}[\Phi(k)+\Phi(n+k)]
                                         \cos\frac{(3n+2k)\pi}{4}
  \end{split}
  \label{equ:Ker2}
\end{equation}
```

La differenza fra *split* e *aligned* è che il primo ambiente, oltre a consentire allineamenti verticali, come fa anche il secondo ambiente, produce un risultato finale che, a parte il suo contenuto, ha esattamente la larghezza della riga in cui compare, mentre *aligned* produce il suo risultato finale sotto forma di un oggetto che è largo quanto il suo contenuto. In tal modo questo oggetto può costituire un elemento di costruzione di un oggetto più grande. Questi due ambienti, inoltre, differiscono dall'ambiente *multline* perché consentono degli allineamenti verticali, mentre *multline* non lo consente, come si vede ne paragrafo seguente.

14.4.4 L'ambiente *multline*

La stessa equazione su tre righe composta con l'ambiente *multline* diventa:

$$\begin{aligned}\mathrm{Ker}_n(x) = &-[\log(x/2)+\gamma] + \tfrac{1}{4}\pi\,\mathrm{Bei}_n(x)\\ &+\frac{1}{2}\sum_{k=0}^{n-1}\frac{(n-k-1)!(x/2)^{2k-n}}{k!}\cos\frac{(3n+2k)\pi}{4}\\ &+\frac{1}{2}\sum_{k=0}^{\infty}\frac{(x/2)^{n+2k}}{k!(n+k)!}[\Phi(k)+\Phi(n+k)]\cos\frac{(3n+2k)\pi}{4}\end{aligned}\tag{14.7}$$

con il codice sorgente:

```
\begin{multline}
  \Ker_n(x)  = -[\log(x/2)+\gamma]
                       +{\textstyle\frac{1}{4}}\pi\Bei_n(x)\\
  +\frac{1}{2}\sum_{k=0}^{n-1}
               \frac{(n-k-1)!(x/2)^{2k-n}}{k!}
                                    \cos\frac{(3n+2k)\pi}{4}\\
  +\frac{1}{2}\sum_{k=0}^\infty
       \frac{(x/2)^{n+2k}}{k!(n+k)!}[\Phi(k)+\Phi(n+k)]
                                      \cos\frac{(3n+2k)\pi}{4}
\end{multline}
```

Si noti in particolare la mancanza dei segni di incolonnamento `&` e la mancanza dei comandi di spaziatura `\quad` e `\qquad` presenti negli esempi precedenti; infatti `multline` compone la prima riga allineata con il margine sinistro delle equazioni; allinea l'ultima espressione con il margine destro tenendo conto del numero dell'equazione; infine allinea tutte le altre righe intermedie in modo da centrarle nello spazio disponibile. Se il numero dell'equazione è a destra, è in linea con l'ultima riga, oppure sotto all'ultima riga, se non ci fosse sufficiente spazio; se si decide di comporre con i numeri delle equazioni a sinistra, il numero è allineato con la prima riga oppure è collocato prima della prima riga; in entrambi i casi non c'è mai sovrapposizione fra il numero e le espressioni.

Questa gestione intelligente della collocazione del numero dell'equazione, in modo da non sovrapporsi mai all'equazione stessa, è comune a tutti gli ambienti matematici numerabili prodotti da *amsmath*; questa gestione intelligente già da sola basterebbe a giustificare l'uso sistematico di *amsmath* e dei suoi compagni ogni qual volta si debbano scrivere testi con equazioni anche semplici (prive di segni insoliti) ma lunghe.

È necessario richiamare l'attenzione su come si dividano le espressioni matematiche fra più righe; il punto di cesura va messo *prima* di un operatore di relazione o di un operatore binario, senza ripeterlo alla fine della riga precedente, come è già stato detto. Ma se da un lato non sarebbe opportuno spezzare una espressione di "grandi" dimensioni racchiusa fra parentesi, talvolta è necessario farlo. In questo modo diventa difficile usare correttamente i comandi

`\left` e `\right` per rendere automatico il dimensionamento dei delimitatori, proprio a causa dell'espressione spezzata fra più righe. È quindi necessario rinunciare agli automatismi offerti da LaTeX e ricorrere ai comandi manuali `\bigl`, `\bigr`, eccetera, come mostrato nella pagina 314. Anche se gli ambienti di *amsmath* aiutano moltissimo per comporre espressioni matematiche spezzate, sta all'autore oltre che al "tastierista" trovare il giusto ritmo di divisione di queste lunghe espressioni e di racchiuderle fra le parentesi di dimensioni corrette.

14.4.5 L'ambiente *gather*

L'ambiente *gather* serve per raccogliere ordinatamente da due a più equazioni una per riga, senza nessun incolonnamento particolare ma centrando le varie equazioni; in un certo senso potrebbe essere visto come una sequenza di ambienti *equation*; con questo ambiente si possono raccogliere le equazioni di Maxwell, per esempio, nella forma:

$$\nabla \times \vec{E} = -\frac{\partial \vec{B}}{\partial t} \tag{14.8}$$

$$\nabla \times \vec{H} = \frac{\partial \vec{D}}{\partial t} + \vec{J} \tag{14.9}$$

$$\nabla \cdot \vec{D} = \rho \tag{14.10}$$

$$\nabla \cdot \vec{B} = 0 \tag{14.11}$$

ottenuta scrivendo

```
\begin{gather}
  \nabla\times\vec{E}=-\frac{\partial\vec{B}}{\partial t}\\
  \nabla\times\vec{H}=
                    \frac{\partial\vec{D}}{\partial t}+\vec{J}\\
  \nabla \cdot\vec{D}= \rho\\
  \nabla \cdot\vec{B}= 0
\end{gather}
```

14.4.6 L'ambiente *align*

Al contrario, usando l'ambiente *align* le stesse equazioni di Maxwell diventano:

$$\nabla \times \vec{E} = -\frac{\partial \vec{B}}{\partial t} \tag{14.12}$$

$$\nabla \times \vec{H} = \frac{\partial \vec{D}}{\partial t} + \vec{J} \tag{14.13}$$

$$\nabla \cdot \vec{D} = \rho \tag{14.14}$$

$$\nabla \cdot \vec{B} = 0 \tag{14.15}$$

ottenute scrivendo:

```
\begin{align}
  \nabla\times\vec{E}&= -\frac{\partial\vec{B}}{\partial t}\\
  \nabla\times\vec{H}&=
                    \frac{\partial\vec{D}}{\partial t}+\vec{J}\\
  \nabla \cdot\vec{D}&= \rho\\
  \nabla \cdot\vec{B}&= 0
\end{align}
```

Si nota che l'unica differenza consiste nel comando di incolonnamento `&` inserito prima di ciascun segno di uguaglianza.

All'interno di un ambiente *align* possono comparire diversi incolonnamenti, non uno solo come nell'esempio precedente; per esempio le equazioni di Maxwell potrebbero venire riorganizzate così:

$$\nabla \times \vec{E} = -\frac{\partial \vec{B}}{\partial t} \qquad \nabla \cdot \vec{D} = \rho \tag{14.16}$$

$$\nabla \times \vec{H} = \frac{\partial \vec{D}}{\partial t} + \vec{J} \qquad \nabla \cdot \vec{B} = 0 \tag{14.17}$$

usando la scrittura:

```
\begin{align}
  \nabla\times\vec{E}&=-\frac{\partial\vec{B}}{\partial t}
                    &\nabla \cdot\vec{D}&=\rho\\
  \nabla\times\vec{H}&=\frac{\partial\vec{D}}{\partial t}+\vec{J}
                    &\nabla \cdot\vec{B}&=0
\end{align}
```

14.4.7 L'ambiente *flalign*

L'ambiente *flalign* differisce dall'ambiente *align* solo per il fatto che le espressioni allineate vengono spostate al massimo all'esterno della riga dove compaiono:

$$\nabla \times \vec{E} = -\frac{\partial \vec{B}}{\partial t} \qquad \nabla \cdot \vec{D} = \rho \tag{14.18}$$

$$\nabla \times \vec{H} = \vec{J} + \frac{\partial \vec{D}}{\partial t} \qquad \nabla \cdot \vec{B} = 0 \tag{14.19}$$

e si vede chiaramente che questo ambiente va meglio quando le equazioni non sono numerate e quando le espressioni sono sufficientemente lunghe; nelle altre circostanze è meglio usare *align*. Se le espressioni sono più di due in ogni riga lo spazio viene usato meglio, ma resta il fatto che i numeri delle equazioni diventano ancora più 'superflui'.

14.4.8 L'ambiente *alignat*

L'ambiente *alignat* è un po' particolare, perché accetta un numero come argomento del comando di apertura che specifica quanti punti di allineamento compaiono in ogni riga; la regola dice: conta il massimo numero di segni `&` che compaiono in ogni riga, aggiungi 1 e dividi per 2. Così l'esempio del manuale diventa:

$$\begin{aligned} x &= y_1 - y_2 + y_3 - y_5 + y_8 - \dots && \text{by eq. (3.21)} && (14.20)\\ &= y' \circ y^* && \text{by eq. (4.1)} && (14.21)\\ &= y(0)y' && \text{by Axiom 1} && (14.22) \end{aligned}$$

prodotto con il codice:

```
\begin{alignat}{2}
  x&= y_1 - y_2 + y_3 -y_5 +y_8 -\dots
                        &\quad & \text{by eq.~(3.21)}\\
   &=y'\circ y^*        &      & \text{by eq.~(4.1)}\\
   &= y(0)y'            &      & \text{by Axiom 1}
\end{alignat}
```

e nel quale si osserva la presenza del comando `\text`; questo comando serve per scrivere del testo all'interno di espressioni matematiche; se questo testo contiene una espressione matematica in linea, questa va esplicitamente racchiusa fra i delimitatori `\(` e `\)`.

14.4.9 L'ambiente *subequations*

L'ambiente *subequations* è particolare perché le equazioni all'interno di questo ambiente vengono numerate con un contatore subalterno così da produrre numeri del tipo (10.22a), (10.22b) eccetera. Per esempio, il sistema di proprietà condizionate:

$$\begin{aligned} \mathbf{Im}[F(\sigma + \mathrm{i}\,0)] &= 0 && \forall \sigma > 0 && (14.23a)\\ F(p) &\neq \infty && \forall \sigma > 0 && (14.23b)\\ \lim_{p \to \mathrm{i}\,\omega_0} (p - \mathrm{i}\,\omega_0) F(p) &= k && \forall \omega_0 \text{ e con } 0 \leq k < \infty && (14.23c)\\ \lim_{p \to \infty} F(p)/p &= h && \text{con } 0 \leq h < \infty && (14.23d)\\ \mathbf{Re}[F(0 + \mathrm{i}\,\omega)] &\geq 0 && \forall \omega && (14.23e) \end{aligned}$$

è ottenuto con il codice seguente:

```
\DeclareMathOperator{\uimm}{\mathrm{i}}          % nel preambolo
...
\begin{subequations}
                                                 \label{equ:sube}
  \begin {align}
```

```
    \Im[F(\sigma+\uimm 0)]  &= 0     &\forall\sigma>0\\
    F(p)                    &\not=\infty&\forall\sigma>0\\
    \lim_{p\to\uimm\omega_0}(p-\uimm\omega_0)F(p)&=k
                            & \forall\omega_0\text{ e con }
                                          0\le k<\infty\\
    \lim_{p\to\infty} F(p)/p&=h      &\text{con }0\le h<\infty\\
    \Re[F(0+\uimm\omega)]   &\ge 0   &\forall\omega
                                            \label{equ:sube5}
 \end{align}
\end{subequations}
```

Con il comando `\label` inserito prima dell'ambiente *align*, ma dentro all'ambiente *subequations* si recupera il numero dell'intero sistema di condizioni 14.23; invece con il comando `\label` inserito all'interno dell'ambiente *align* si recupera il 'numero' assegnato a ciascuna condizione, per esempio all'ultima condizione 14.23e si fa riferimento mediante `\ref{equ:sube5}`.

La definizione dell'unità immaginaria `\uimm` è un po' particolare nel senso che essa è definita come un operatore, invece che come una semplice variabile. Si noti che se si usasse il comando primitivo `\mathop`, la definizione dovrebbe contenere oltre alla 'i' in tondo anche uno strut o dello spazio matematico al fine di centrare la 'i' rispetto al suo asse matematico. È questa una ragione in più per ricorrere ai comandi di definizione disponibili con il pacchetto *amsmath*, che permettono di concentrarsi sul significato intrinseco di ciò che si sta definendo, senza badare ai dettagli compositivi a livello di base.

Esercizio 14.1 Il lettore provi a ripetere la composizione di questo sistema di condizioni 14.23 definendo il comando `\uimm` con `\newcommand` ed usando solamente `\mathop`. Che cosa succede?

14.4.10 Gli ambienti spezzati

Di default il pacchetto *amsmath* fa sì che gli ambienti di incolonnamento precedentemente descritti non vengano spezzati da eventuali salti di pagina, ma restino composti interamente in una sola pagina. Questo in generale è preferibile rispetto a quando si è obbligati a voltare pagina per completare la lettura di un gruppo di equazioni raccolte assieme in un uno di questi ambenti.

Tuttavia talvolta è impossibile per vari motivi mantenere tutte le espressioni di un ambiente nella stessa pagina; è sicuramente obbligatorio quando si usano gli ambienti *split*, *aligned*, *gathered*, *alignedat* dove andare ad una nuova pagina è impossibile, mentre con *multline* sarebbe possibile, ma sarebbe anche "sconveniente". Negli altri casi si può essere più tolleranti. Per questo scopo il pacchetto *amsmath* mette a disposizione due comandi:

```
\allowdisplaybreaks[⟨codice⟩]
\displaybreak[⟨codice⟩]
```

per consentire di eseguire un fine pagina fra una espressione e l'altra.

Precisamente `\allowdisplaybreaks` può essere messo nel preambolo e vale per tutti gli incolonnamenti matematici del documento; se all'interno di uno di questi ambienti *non* si vuole eseguire un cambio di pagina, invece di usare il comune comando `\\` si può usare la sua variante asteriscata `\\*` che, appunto impedisce il salto di pagina.

Invece il comando `\displaybreak` va usato solamente prima del comando `\\` dopo il quale si desidera consentire un salto di pagina.

Il ⟨*codice*⟩ per entrambi i comandi funziona alla stessa maniera dei comandi simili `\pagebreak` da usarsi nel corpo del testo. Si tratta di una cifra nell'intervallo `0`–`4`[2] che dà un peso alla "licenza" di eseguire un cambio di pagina: `0` è un blando consenso, mentre `4` è un ordine perentorio.

Vale la pena di osservare qui che il comando di fine espressione `\\` ammette la stessa sintassi che esso usa quando viene usato nel testo (vedi la sintassi esposta nella pagina 711), sia per la versione normale, sia per quella asteriscata:

```
\\[⟨spazio⟩]
\\*[⟨spazio⟩]
```

cosicché, usando il valore ⟨*spazio*⟩, si può controllare anche la spaziatura fra le equazioni.

14.5 Altri comandi e ambienti

Il pacchetto *amsmath* offre numerosi altri comandi e ambienti che consentono di comporre la matematica in modo estremamente professionale.

14.5.1 Definizione di operatori funzionali

Qui si cita nuovamente il comando per definire i nomi di nuovi operatori funzionali:

```
\DeclareMathOperator{⟨operatore⟩}{⟨definizione⟩}
```

così che se si volesse definire l'operatore per l'arcoseno iperbolico 'arsinh' (che non è definito né da LaTeX standard, né da `amsmath`) basterebbe dichiarare nel preambolo:

```
\DeclareMathOperator{\arsinh}{arsinh}
```

Il comando asteriscato serve per definire operatori funzionali con i limiti.

[2] Solo il comando `\allowdisplaybreak` non ammette il valore `0`.

14.5.2 Le frazioni in generale e le frazioni continue

14.5.2.1 Le frazioni e gli altri costrutti simili

Il pacchetto *amsmath* consente di scrivere le frazioni e gli altri costrutti simili alle frazioni in tre varianti distinte e quindi con tre comandi distinti: il comando 'normale' che compone in uno stile diverso in modo testo rispetto al modo display, il nome testuale per comporre sempre e comunque in modo testo, e il modo display per scrivere sempre e comunque in modo display; per esempio le frazioni normalmente vengono composte con il comando `\frac` ma il modo testuale viene composto con `\tfrac` e il modo display con `\dfrac`. L'operatore binomiale, per esempio si compone con `\binom`:

$$\binom{n}{k} = \frac{n(n-1)(n-2)\cdots(n-k+1)}{1\cdot 2\cdot 3\cdots k}$$

Se proprio si volesse indicare il coefficiente binomiale in modo display all'interno del testo, e si usasse `\dbinom`, si otterrebbe $\dbinom{n}{k}$; come si vede, questo è un tipo di scrittura da evitare assolutamente quando si compone matematica in linea con il testo, perché altera vistosamente lo scartamento.

14.5.2.2 Le frazioni continue

Fra i comandi utilissimi conviene citare quello per la composizione delle frazioni continue `\cfrac` con il quale si possono comporre facilmente espressioni del tipo:

$$\tanh z = \cfrac{z}{1+\cfrac{z^2}{3+\cfrac{z^2}{5+\cfrac{z^2}{7+\cfrac{z^2}{\cdots}}}}} \tag{14.24}$$

composta con i comandi:

```
\begin{equation}
  \tanh z = \cfrac{z}{1+\cfrac{z^2}{3+\cfrac{z^2}{5+
                       \cfrac{z^2}{7+\cfrac{z^2}{\cdots}}}}}
\end{equation}
```

14.5.3 Il testo intercalato alle equazioni

Un altro comando molto interessante è `\intertext`; esso consente di inserire un (breve) testo che interrompe un incolonnamento di equazioni senza perdere i

parametri dell'incolonnamento, così che le successive equazioni mantengono lo stesso incolonnamento; per esempio:

$$a_4x^4 + a_2x^2 + a_0 = 0 \tag{14.25}$$

diventa una semplice equazione di secondo grado:

$$a_4t^2 + a_2t + a_0 = 0 \tag{14.26}$$

se si pone:

$$t = x^2 \tag{14.27}$$

Simile a `\intertext` è il comando `\text`, già visto; esso serve per inserire un po' di testo dentro una formula, specialmente negli enunciati delle condizioni, ma non solo; esso serve anche e tipicamente all'interno dell'ambiente *cases*; l'equazione 13.5 usata per mostrare la composizione dei delimitatori estensibili e del delimitatore invisibile, può essere scritta mediante questo ambiente e questo comando nella maniera seguente:

$$x_{1,2} = \begin{cases} \dfrac{-b \pm \sqrt{b^2 - 4ac}}{2a} & \text{se } b^2 - 4ac > 0 \\ -\dfrac{b}{2a} & \text{se } b^2 - 4ac = 0 \\ \dfrac{-b \pm \mathrm{i}\sqrt{4ac - b^2}}{2a} & \text{se } b^2 - 4ac < 0 \end{cases} \tag{14.28}$$

scrivendo nel file sorgente:

```
  \begin{equation}
  x_{1,2}=
  \begin{cases}
    \dfrac{-b\pm\sqrt{b^2-4ac}}{2a}
                              &\text{se \(b^2-4ac>0\)}\\[1.5ex]
    -\dfrac{b}{2a}            &\text{se \(b^2-4ac=0\)}\\[1.5ex]
    \dfrac{-b\pm\uimm\sqrt{4ac-b^2}}{2a}
                              &\text{se \(b^2-4ac<0\)}
  \end{cases}
\end{equation}
```

Si noti che nell'ambito dell'argomento di `\text` la condizione è nuovamente composta come matematica in linea, quindi è necessario usare gli appositi comandi `\(` e `\)` come si fa normalmente quando si compone all'interno dei capoversi. Si noti anche l'uso del comando `\dfrac` per obbligare LaTeX a comporre le frazioni in modo display, nonostante che spontaneamente esso comporrebbe quelle frazioni in modo testo; l'ulteriore spaziatura mediante l'argomento facoltativo `\\[1.5ex]` serve per compensare l'effetto prodotto dall'uso delle frazioni in display.

Esercizio 14.2 Che cosa succederebbe se nell'equazione 14.28 non si usasse `\dfrac`?

Esercizio 14.3 Che cosa succederebbe nell'equazione 14.28 se si usasse `\\` senza il suo argomento facoltativo?

14.5.4 Le frecce estensibili

Le frecce estensibili diventano lunghe quanto i loro argomenti; si tratta dei comandi `\xleftarrow` e `\xrightarrow` che accettano due argomenti, uno obbligatorio che sarà collocato sopra la freccia e uno facoltativo che verrà collocato sotto la freccia:

$$A \xleftarrow{n+\mu-1} B \xrightarrow[T]{n\pm i-1} C \tag{14.29}$$

composto con

```
A \xleftarrow{n+\mu-1} B \xrightarrow[T]{n\pm i-1} C
```

14.5.5 Gli indici incolonnati

Talvolta può essere necessario indicare gli indici di somma o di prodotto con diverse indicazioni o condizioni, come in

$$\sum_{\substack{0\le i\le m\\ 0<j<n}} P(i,j) \tag{14.30}$$

Questo risultato si ottiene mediante `\substack` come si vede dall'esempio 14.30, composto con

```
\begin{equation}
\sum_{\substack{0\le i\le m\\
                0<j<n
               }
     } P(i, j)
\end{equation}
```

14.5.6 Gli integrali multipli

Nel caso di integrali multipli LaTeX senza estensioni necessiterebbe di un aggiustamento della spaziatura; si osservi:

$$\int\int\int\int \ldots \mathrm{d}w\,\mathrm{d}x\,\mathrm{d}y\,\mathrm{d}z \qquad \iiiint \ldots \mathrm{d}w\,\mathrm{d}x\,\mathrm{d}y\,\mathrm{d}z \tag{14.31}$$

Nel primo caso si è usato quattro volte il segno di integrale, mentre nel secondo caso si è usato il comando per l'integrale quadruplo disponibile con il pacchetto *amsmath* che mette a disposizione anche i comandi per l'integrale doppio e per quello triplo; in totale i comandi disponibili sono `\int` per l'integrale semplice, `\iint` per l'integrale doppio, `\iiint` per l'integrale triplo e `\iiiint` per l'integrale quadruplo; infine c'è anche `\idotsint` per l'integrale multiplo.

14.5.7 L'operatore differenziale

Si noti nell'equazione 14.31 il segno di differenziale usato come un operatore speciale che lascia il suo normale spazio alla sua sinistra ma non lascia spazio alla sua destra; inoltre esso è scritto in tondo come prescrivono le norme ISO valide per le espressioni matematiche per la fisica e le altre scienze sperimentali. La matematica non considera il segno di differenziale come un operatore. Qui lo chiamiamo operatore speciale per giustificare perché usiamo il comando `\mathop` per definirlo. È chiaro che non si tratta di un normale operatore definibile con il comando già descritto `\DeclareMathOperator`, perché con questa definizione il differenziale avrebbe lo spazio anche alla sua destra. Nel preambolo è stato invece definito la macro per il differenziale `\diff` servendosi invece di un operatore 'vuoto', così:

```
\newcommand*{\diff}{\mathop{}\!\mathrm{d}}
```

I dettagli di `\newcommand` verranno visti nel capitolo 19; qui limitiamo la nostra attenzione sull'operatore `\mathop` il cui argomento viene definito come operatore matematico, salvo che in questo caso il suo argomento è vuoto; nonostante ciò gli spazi che attorniano gli operatori sono rispettati, anche a destra; ma questo spazio a destra non ci vuole, quindi con il comando `\!` si arretra della stessa quantità, dopo di che si inserisce il segno del differenziale in tondo. Ecco, questa è una delle rare volte in cui si è usato un comando di spaziatura matematica (arretramento di uno spazio fine), ma lo si è fatto a ragion veduta per definire un comando per un simbolo speciale come il differenziale.

Come già detto, i matematici affermano che il simbolo del differenziale non rappresenta un operatore; qui non si discute la sua intima essenza, ma si sottolinea il fatto che le norme ISO prescrivono la scrittura in tondo solo per le espressioni matematiche della fisica e delle altre scienze sperimentali, dove le variabili devono essere scritte in corsivo e dove la maggior parte delle variabili ha significati standardizzati. Nel caso specifico d può voler dire: diametro, densità, distanza dei piani reticolari, coefficiente piezoelettrico, eccetera, come si deduce dalla tabella 24.9 del paragrafo 24.2. La possibile confusione del segno corsivo del differenziale con uno di quei simboli standardizzati giustifica la norma ISO, almeno agli occhi dei fisici e dei tecnologi.

Vale la pena di sottolineare che gli spazi del differenziale sono necessari per scrivere gli integrali, mentre servono di meno per scrivere le equazioni differenziali, ma ci pensa LaTeX a tenerli nella debita considerazione, come nella seguente equazione:

$$a\frac{\mathrm{d}^2x}{\mathrm{d}t^2} + b\frac{\mathrm{d}x}{\mathrm{d}t} + c = f(t)$$

scritta usando il codice:

```
\[
a\frac{\diff^2x}{\diff t^2} + b\frac{\diff x}{\diff t} + c = f(t)
\]
```

14.5.8 I simboli corsivi matematici in nero

Una cosa che capita spesso in matematica è quella di usare il carattere nero all'interno di una formula composta con carattere medio. LaTeX senza estensioni non consente di scegliere altro che una formula tutta in carattere medio oppure tutta in carattere nero, non consente di mescolare questi due casi, a meno di non fare acrobazie di tipo prescrittivo, piuttosto che di mark-up. Inoltre se si desidera comporre una formula tutta nera bisogna dichiararlo prima di entrare in modo matematico, e di conseguenza occorre proteggere la formula 'nera' dentro un gruppo, oppure è necessario cancellare l'effetto di annerimento mediante l'apposito comando:

```
\boldmath
⟨ambiente matematico nero⟩
\unboldmath
```

All'interno di una espressione è possibile usare il carattere nero, ma solo per il tondo, non per il corsivo matematico. Il comando sarebbe `\mathbf` del tutto analogo al corrispondente comando per il testo `\textbf`; ma per il corsivo matematico non c'è nulla.

L'estensione fornita da *amsmath* consente di usare il comando `\boldsymbol` il cui argomento può essere qualunque simbolo letterale o operazionale matematico:

```
\boldsymbol{⟨simbolo matematico⟩}
```

a cui si aggiunge una scorciatoia (anche qualitativa) del *poor man bold*, `\pmb`, che simula il font nero replicando un segno chiaro tre volte con un leggero spostamento a destra e in alto (e questo rende meno definito il segno; ecco perché qualitativamente si tratta di una soluzione da pover'uomo). La sintassi è la stessa:

```
\pmb{⟨simbolo matematico⟩}
```

In generale si sconsiglia di usare `\pmb`; con i font di serie con LaTeX non dovrebbe essere necessario usarlo altro che per i grandi operatori e per i delimitatori estensibili, ma non sembra una operazione molto frequente da fare.

Con `\boldsymbol` è possibile, per esempio, usare il corsivo matematico per indicare i vettori:

$$\boldsymbol{a} = a_x\boldsymbol{i} + a_y\boldsymbol{j} + a_z\boldsymbol{k}$$

mentre con `\pmb` si ottiene:

$$\pmb{a} = a_x\pmb{i} + a_y\pmb{j} + a_z\pmb{k}$$

e la differenza si vede ad occhio nudo.

È disponibile sempre anche il pacchetto *bm* che produce lo stesso effetto ma in modo più efficiente; per compatibilità con *amsmath* ridefinisce `\boldsymbol`

in modo che sia completamente equivalente al suo comando specifico `\bm`. Questo pacchetto, però, è indipendente da *amsmath* e può essere usato con o senza il secondo.

14.5.9 Le espressioni matematiche riquadrate

Secondo lo scrivente per mettere in risalto una espressione matematica si può usare il meccanismo fornito da LaTeX, cioè mettendola in display; talvolta, però, potrebbe essere necessario ricorrere ad un'enfasi maggiore e per questo scopo basta racchiudere l'espressione dentro ad una cornice; *amsmath* mette a disposizione il comando `\boxed` come nell'esempio seguente:

$$\boxed{x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}}$$

composto con

```
\[
\boxed{x_{1,2}= \frac{-b\pm\sqrt{b^2-4ac}}{2a}}
\]
```

14.6 Le matrici e i determinanti

Le matrici e i determinanti sono particolari tabelle che contengono solo, o prevalentemente, espressioni matematiche; *amsmath* mette a disposizione un certo numero di ambienti per comporre matrici variamente delimitate e perciò con significati matematici diversi.

L'ambiente base è ovviamente *matrix* e produce una matrice avente al massimo 10 colonne, ma sprovvista di delimitatori; anche gli altri ambienti consentono un massimo di 10 colonne; questo numero può venire cambiato se si imposta il contatore `MaxMatrixCols` con un valore diverso:

`\setcounter{MaxMatrixCols}{`⟨*numero delle colonne*⟩`}`

Gli altri ambienti per costruire matrici o determinanti sono i seguenti: *pmatrix*, *bmatrix*, *Bmatrix*, *vmatrix* e *Vmatrix*; tutti questi racchiudono la matrice vera e propria all'interno di una coppia di delimitatori estensibili; a seconda del delimitatore la matrice ha un significato diverso. Gli ambienti *pmatrix*, *bmatrix* e *Bmatrix* racchiudono la matrice all'interno di una coppia di parentesi tonde, oppure quadre, oppure graffe; l'ambiente *vmatrix* serve per comporre un determinante racchiudendo la matrice fra due aste verticali, mentre l'ambiente *Vmatrix* serve per comporre una norma racchiudendo la matrice fra due coppie di barre verticali.

Grazie a questi ambienti non è difficile comporre una espressione del tipo:

$$\begin{vmatrix} 1 & 1 & 0 & 0 \\ -1 & 2 & 1 & 0 \\ 0 & -1 & 3 & 1 \\ 0 & 0 & -1 & 4 \end{vmatrix} = 30 \tag{14.32}$$

mediante il codice:

```
\begin{equation}
    \begin{vmatrix}
      1 & 1 & 0  & 0 \\
     -1 & 2 & 1  & 0 \\
      0 &-1 & 3  & 1 \\
      0 & 0 &-1  & 4
    \end{vmatrix}
    = 30
\label{equ:determinante}
\end{equation}
```

La scrittura nel file sorgente del codice per la composizione della matematica è più facilmente leggibile se si usa una scrittura strutturata; ci si rende più facilmente conto se ogni segno di apertura è in corrispondenza con il suo compagno di chiusura; usando questa pratica, nei limiti del possibile si evitano moltissimi errori e si compone più speditamente.

Una equazione matriciale diventa abbastanza semplice da comporre, specialmente se si ha l'accortezza di scrivere in modo strutturato nel file sorgente tutto quello che va ordinatamente inserito nelle matrici:

$$\begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} \\ a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} \\ a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} \\ a_{4,1} & a_{4,2} & a_{4,3} & a_{4,4} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{pmatrix}$$

Il codice sorgente è:

```
\[
  \begin{pmatrix}
    a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} \\
    a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} \\
    a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} \\
    a_{4,1} & a_{4,2} & a_{4,3} & a_{4,4}
  \end{pmatrix}
  \begin{pmatrix}
    x_1\\x_2\\x_3\\x_4
  \end{pmatrix}
  =
```

```
  \begin{pmatrix}
    y_1\\y_2\\y_3\\y_4
  \end{pmatrix}
\]
```

Tuttavia esistono certe necessità della matematica che non possono essere soddisfatte con i mezzi messi a disposizione dal pacchetto *amsmath*, come per esempio quella di ripartire una matrice in sottomatrici mediante linee tratteggiate, come nell'equazione 14.33 ottenuta usando il pacchetto *tabu* e il suo ambiente *tabu* in ambiente matematico, quindi in sostituzione dell'ambiente *array*.

$$\left(\begin{array}{ccc:ccc} a_{1,1} & \cdots & a_{1,n} & b_{1,n+1} & \cdots & b_{1,n+m+1} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & \cdots & a_{n,n} & b_{n,n+1} & \cdots & b_{n,n+m+1} \\ \hdashline c_{n+1,1} & \cdots & c_{n+1,n} & d_{n+1,n+1} & \cdots & d_{n+1,n+m+1} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ c_{n+m+1,1} & \cdots & c_{n+m+1,n} & d_{n+m+1,n+1} & \cdots & d_{n+m+1,n+m+1} \end{array}\right) \tag{14.33}$$

Prima di comporre l'equazione 14.33, possibilmente dentro un gruppo che contenga anche l'equazione, si definisce lo stile delle linee tratteggiate orizzontali e verticali con il comando `\tabulinestyle`:

```
\tabulinestyle{on 2pt off 3pt}
```

dove la stringa `on` indica la lunghezza dei tratti "pieni" e la stringa `off` quella dei tratti "vuoti"; in realtà la sintassi di `\tabulinestyle` è più complessa e consente di specificare anche lo spessore della linea tratteggiata e il colore dei tratti pieni e vuoti; si veda la documentazione del pacchetto *tabu*. Il comando `\tabucline` può ricevere l'indicazione delle colonne che deve sottolineare (un po' come il comando `\cline` del nucleo di LaTeX), ma con la scrittura `\tabucline-` sottolinea tutte le colonne presenti nella tabella o nella matrice. Le linee verticali sono invece indicati dai soliti indicatori nell'argomento del comando di apertura dell'ambiente *tabu*. Ciò premesso, diventa chiaro il codice con il quale si è composta l'equazione 14.33:

```
{\tabulinestyle{on2pt off3pt}
\tabulinesep=1.5mm
\begin{equation}
\left(
\begin{tabu}{@{}*3c|*3c@{}}
a_{1,1}    & \cdots  & a_{1,n}   & b_{1,n+1}     & \cdots & b_{1,n+m+1}     \\
\vdots     & \ddots  & \vdots    &  \vdots       & \ddots & \vdots          \\
a_{n,1}    & \cdots  & a_{n,n}   & b_{n,n+1}     & \cdots & b_{n,n+m+1}     \\
\tabucline-
```

```
c_{n+1,1}  &  \cdots & c_{n+1,n} & d_{n+1,n+1}   & \cdots & d_{n+1,n+m+1}   \\
\vdots     &  \ddots & \vdots    &  \vdots       & \ddots & \vdots          \\
c_{n+m+1,1}&  \cdots &c_{n+m+1,n}& d_{n+m+1,n+1} & \cdots & d_{n+m+1,n+m+1} \\
\end{tabu}
\right)
\end{equation}}
```

Si sottolinea il fatto che che i "filetti" di separazione fra righe e/o colonne della tabella o della matrice sono solitamente poco professionali in tipografia; l'uso qui fatto per separare la matrice in sottomatrici invece è perfettamente giustificato.

14.7 I diagrammi commutativi

I diagrammi commutativi sono particolari espressioni matematiche a metà strada fra un insieme di espressioni matematiche e un disegno che mette in relazione le une con le altre.

Il pacchetto *amscd* serve per estendere *amsmath* definendo un ambiente *CD* con il quale è possibile disegnare diagrammi commutativi piani e senza frecce diagonali. Per diagrammi più complessi il lettore deve rivolgersi a pacchetti di estensione più specializzati come *xypic* oppure *kuvio*, senza trascurare l'onnipotente pacchetto *tikz*.

Il diagramma commutativo:

$$\begin{array}{ccc} S^{\mathcal{W}_\Lambda}\otimes T & \xrightarrow{\ j\ } & T \\ \big\downarrow & & \big\downarrow{\scriptstyle \text{End } P} \\ (S\otimes T)/I & = & (Z\otimes T)/J \end{array}$$

è stato composto con il codice:

```
\[
  \begin{CD}
    S^{\mathcal{W}_\Lambda}\otimes T @>j>> T \\
    @VVV                             @VV{\mathrm{End}\:P}V\\
    (S\otimes T)/I                   @=(Z\otimes T)/J
  \end{CD}
\]
```

Oltre ai vari comandi per inserire le frecce o le linee di collegamento fra le varie espressioni, si noti l'uso di `\mathcal` per scrivere caratteri 'calligrafici' e l'uso del comando di spaziatura `\:` per inserire uno spazio circa pari a quello interparola; il comando `\:` in matematica si comporta diversamente dai comandi analoghi per il modo testo `\enspace` o `\thinspace`, perché esso cambia valore a seconda che l'espressione matematica che si sta componendo sia in modo testo, in modo display, in modo degli indici del primo ordine o di quelli del secondo ordine.

Si prenda nota anche del fatto che l'ambiente *CD* deve essere realizzato all'interno di un ambiente matematico come quello prodotto da `\[...\]` oppure uno dei vari altri ambienti visti in questo capitolo.

A proposito di *xypic* conviene sottolineare che esso usa una sintassi molto particolare dove si fa uso di caratteri speciali per eseguire certe formattazioni; uno di questi caratteri è il segno `"` che con l'opzione *italian* di *babel*[3] è attivo e svolge altre funzioni diverse da quelle che esso è chiamato a svolgere quando si usa *xypic*. Per questo conviene ridefinire il comando di apertura dell'ambiente *xy* nel modo seguente:

```
\toks0=\expandafter{\xy}
\edef\xy{\noexpand\shorthandoff{\noexpand"}\the\toks0 }
```

Si tratta di programmazione con il linguaggio primitivo di TEX, salvo per il comando `\shorthandoff` che, insieme al suo compagno `\shorthandon`, serve per disattivare o, rispettivamente, per riattivare uno o più caratteri attivi; è un comando definito da *babel*. In questo modo, se si usa il pacchetto *xypic*, basta dare quelle definizioni subito dopo la chiamata del pacchetto e poi si è sicuri di poter usare tranquillamente l'ambiente *xy* senza i problemi che il carattere attivo potrebbe dare. Si noti che questo trucco *deve* venire usato ogni volta che si compone in una lingua diversa dall'inglese, perché in tutte le altre lingue europee il segno `"` viene sempre definito attivo.

14.8 La punteggiatura in matematica

La punteggiatura in matematica è tipograficamente un problema aperto perché non esistono norme che regolino la questione; è più una questione di consuetudini tipografiche ed editoriali. Anzi, ogni casa editrice si premura spesso di fornire ai suoi potenziali autori una breve guida in cui sono indicate le tradizioni compositive della casa editrice. Gli argomenti spaziano attraverso tutto lo scibile che può essere scritto, e vanno dai titoli correnti alle testatine e ai piedini, fino alle virgole del testo e la punteggiatura in matematica; naturalmente se la casa editrice riceve un contributo da pubblicare in forma direttamente stampabile, generalmente fornisce i fogli di stile nel linguaggio di videocomposizione che l'autore userà. Non sono rari i casi di case editrici che forniscono direttamente i file di classe e/o le eventuali estensioni per testi da pubblicare dopo la composizione con LATEX.

Tuttavia per quel che riguarda la punteggiatura della matematica qualche volta le informazioni o le prescrizioni sono lacunose, quindi l'autore deve supplire con la sua esperienza e la sua sensibilità.

[3]Dalla versione 1.3k, il modulo per l'italiano di *babel* definisce le doppie virgolette attive se e solo se dopo aver caricato *babel* e prima di `\begin{document}` si dà il comando `\setactivedoublequote`. Una volta attivate, le doppie virgolette diritte possono essere disattivate e riattivate solo mediante i comando`\shorthandoff` e `\shorthandon`.

Non dovrebbero sussistere problemi per le espressioni matematiche (brevi) che compaiono in linea con il testo, dove la punteggiatura dominante è quella testuale.

Direi che non dovrebbero esserci problemi nemmeno per la punteggiatura *interna* a ogni espressione matematica; questa punteggiatura specifica interna alle espressioni matematiche ha un suo particolare significato matematico che fa parte della sua 'grammatica'. I problemi maggiori nascono per la punteggiatura *esterna* alle espressioni matematiche in display.

Tuttavia cominciamo con i problemi minori di punteggiatura interna. In matematica si usano essenzialmente gli stessi segni di interpunzione che si usano nel testo, eccetto il punto interrogativo, salvo che alcuni di questi segni hanno un significato particolare; per esempio il punto esclamativo che segue un numero intero o un simbolo che rappresenta un numero intero, significa che di quel numero si considera il 'fattoriale':

$$n! = n(n-1)(n-2)\cdots 2\cdot 1$$

Lo stesso punto esclamativo può servire per cambiare il significato di un 'quantificatore' logico; per esempio $\exists!$ da taluni è usato nel senso di 'esiste ed è unico'.

I due punti hanno un significato particolare in espressioni logiche, così come (in Italia e secondo le ultime versioni delle norme ISO 80000) possono indicare l'operazione di divisione, quindi non vengono usati come segno di interpunzione.

Il punto e virgola ha il solo significato di segno di interpunzione, mentre la virgola e il punto sono 'anfoteri', possono servire ora con il significato di segno di interpunzione, ora con il significato di separatore decimale.

Scrivendo in inglese, il punto serve come separatore decimale e come punto fermo, mentre la virgola è un semplice segno di interpunzione, usato, come punteggiatura interna, per separare gli elementi di una lista.

Al contrario, in qualunque lingua diversa dall'inglese le norme ISO prescrivono la virgola nella funzione di separatore decimale, e quindi la virgola viene ad assumere due ruoli distinti; il punto, invece sembrerebbe declassato a svolgere la sola funzione di segno di interpunzione (raro nell'interpunzione interna), ma poi rientra come separatore decimale nelle costanti numeriche 'reali' di tutti i linguaggi di programmazione. Nel capitolo 19 si vedrà come fare per convincere LaTeX a trattare correttamente le due funzioni della virgola in matematica.

La punteggiatura esterna è quella che pone i principali problemi. Mi spiego meglio: per *punteggiatura esterna* intendo sia la punteggiatura che compare in una espressione che contenga al suo interno anche delle parti testuali, sia la punteggiatura finale.

Non esistono regole 'grammaticali' codificate per gestire la punteggiatura esterna. Scrivendo per una data casa editrice, come già detto, è opportuno riferirsi alle sue regole editoriali. Scrivendo per proprio conto e, specialmente, quando si è editori di se stessi, bisogna decidere autonomamente e scegliere uno stile di punteggiatura esterna per la matematica.

Sul forum del GUIT alcuni argomenti sono stati dibattuti ampiamente e uno di questi argomenti trattava appunto della punteggiatura esterna. Sono emerse due scuole di pensiero fra loro antitetiche, ma ciascuna con pregi e difetti; l'unica cosa sulla quale le due scuole di pensiero concordano è che scelto uno stile di punteggiatura, quello deve essere usato costantemente e coerentemente; il lettore non deve essere confuso da un uso irregolare e non uniforme della punteggiatura. Le due scuole di pensiero dicono sostanzialmente quanto segue.

La punteggiatura esterna è necessaria in matematica come lo è per il testo Secondo questo orientamento la punteggiatura separa le varie parti di una espressione che contiene anche dei frammenti testuali e va inserita anche alla fine delle espressioni in display, sia sotto forma di virgole, di punti e virgola o di punti finali; in altre parole l'espressione matematica, anche se è in display, fa parte del testo e la punteggiatura ne esprime il ritmo esattamente come lo fa quando si compone del testo non matematico. Bisogna solo stare attenti a che i segni di punteggiatura non interferiscano con la parte matematica in senso stretto al punto da renderne difficile la lettura o, peggio ancora, al punto di indurre il lettore a farne una lettura ambigua o errata.

Siccome la punteggiatura non va staccata da quanto viene prima (a differenza del francese, nel quale almeno il punto e virgola va spaziato con uno spazio di almeno un terzo di em da quanto viene prima), talvolta in italiano si potrebbero spaziare virgole e punti, suscettibili di introdurre ambiguità nella lettura delle espressioni, mediante uno spazio fine, che in matematica si rende semplicemente con il comando `\,`; si confronti, per esempio,

$$w = \frac{az+b}{cz+d},\ \text{con } a,b,c,d \in \mathbb{R};\ z,w \in \mathbb{C}$$

con

$$w = \frac{az+b}{cz+d}\,,\ \text{con } a,b,c,d \in \mathbb{R};\ z,w \in \mathbb{C}.$$

Si vede benissimo che nel secondo caso la virgola non può essere confusa con un apice della variabile d.

La matematica si scrive senza punteggiatura esterna Come si vede in questo stesso capoverso, LaTeX sostituisce la punteggiatura finale del titolino (composto con il comando `\paragraph`) con uno spazio interparola maggiore. La scuola di pensiero che non usa la punteggiatura esterna è in linea con la pratica tipografica di usare spazi adeguati al posto della punteggiatura. La punteggiatura esterna nelle espressioni matematiche viene sostituita con degli spazi o degli incolonnamenti: se si deve separare un frammento testuale all'interno di una espressione, lo si fa usando spazi di almeno un quadrato, ma meglio ancora di un quadratone, rispettivamente con i comandi `\quad` e `\qquad`; un incolonnamento eseguito per esempio con l'ambiente *align* permette di separare le colonne delle condizioni dalle colonne contenenti espressioni matematiche, come si è fatto, per

esempio, con il sistema di condizioni 14.23. Proseguendo l'esempio precedente si esamini ancora

$$w = \frac{az + b}{cz + d} \qquad \text{con } a, b, c, d \in \mathbb{R} \quad z, w \in \mathbb{C}$$

Secondo questa scuola di pensiero la punteggiatura finale è ridondante, perché gli spazi prima e dopo una formula in display e il modo di scrivere la prima riga di testo che segue tale formula sono sufficienti a capire se la formula stessa chiude un capoverso, o se il periodo di cui fa parte finisce con la formula, oppure se continua dopo la formula; infatti se questa prima riga comincia con una lettera minuscola (e, ovviamente, non è rientrata) significa che il periodo continua dopo la formula; se questa prima riga non è rientrata ma comincia con una lettera maiuscola, la formula termina il periodo, e ne comincia uno nuovo subito dopo, ma il capoverso non è terminato; se invece questa prima riga comincia con una lettera maiuscola ed è rientrata, vuol dire che dopo la formula, che chiude il periodo, comincia un nuovo capoverso. Quello che l'assenza di punteggiatura non permette di capire è il quarto caso, cioè se la formula termina una frase o se la frase continua dopo la formula; ma anche senza punteggiatura il senso del discorso è evidente anche in questo caso per la presenza di congiunzioni o altri connettivi che permettono di leggere l'espressione in display come ultimo elemento di una frase o come elemento interno ad una frase. In questi quattro casi l'altra scuola di pensiero avrebbe usato rispettivamente la virgola, il punto fermo, il punto e a capo (di fatto il punto alla fine della formula) oppure il punto e virgola.

Inoltre, con riferimento all'espressione 14.28, dove si sarebbe messa la virgola di chiusura dell'espressione? In casi analoghi si è vista la virgola (o altri segni di interpunzione) allineata con l'asse matematico[4] dell'espressione, oppure a chiusura dell'ultima riga, ma non è semplice prendere una decisione compositiva in situazioni del genere.

L'approccio seguito dalla scuola di pensiero dell'assenza di punteggiatura esterna è coerente con le regole formulate per le elencazioni semplici, dove è normale non usare la punteggiatura alla fine delle singole voci, ma è normale usare una scrittura *ellittica* nelle elencazioni semplici, mentre è da evitare assolutamente nelle enumerazioni. Quello delle elencazioni e delle enumerazioni ha a che fare solo con il testo, ma aver specificato che ogni elencazione semplice fa parte del capoverso che la precede, fa capire perché è consentita una forma di scrittura ellittica, che invece è vietata nelle enumerazioni, dove ogni voce è formata da almeno un capoverso completo. Mescolando testo e matematica non si possono ripetere gli stessi ragionamenti; tuttavia la forma ellittica per la matematica ha la sua giustificazione.

[4] L'asse matematico è la linea orizzontale (ideale, quindi invisibile) che passa a metà fra le due lineette che formano il segno =; gli operatori, le frazioni e tanti altri elementi delle espressioni matematiche dovrebbero avere un ingombro verticale simmetrico rispetto a questo asse.

Uso simultaneo della punteggiatura e degli spazi in matematica Naturalmente nessuno vieta di usare sia la punteggiatura sia gli spazi; l'esempio riportato sopra diventerebbe

$$w = \frac{az + b}{cz + d}\,, \qquad \text{con } a, b, c, d \in \mathbb{R}; \quad z, w \in \mathbb{C}.$$

Come il lettore avrà notato, in questo testo si segue la seconda linea di pensiero; è una scelta del coordinatore, il quale si prende la responsabilità di questa scelta, ed è perfettamente conscio che i collaboratori seguaci dell'altra scuola di pensiero ne saranno un po' delusi. L'omogeneità dello stile e, quindi, la coerenza compositiva, richiedono di non cambiare la punteggiatura da un paragrafo all'altro, a seconda di chi l'ha scritto. Il lettore invece, quando compone i suoi testi, seguirà le sue convinzioni e sceglierà il tipo di punteggiatura che preferisce.

Nell'ultima versione de *L'Arte di scrivere con LaTeX*, [55], gli autori raccomandano di non usare la punteggiatura esterna.

Va infine segnalato il documento *Preparing Articles with LaTeX — Instructions to authors for preparing compuscripts* della casa editrice olandese Elsevier [22], ben nota in campo scientifico; si tratta delle istruzioni per gli autori e, per quel che riguarda la matematica, non è espressamente detto nulla sulla punteggiatura, ma gli esempi in display ne sono tutti privi.

Vale la pena di segnalare anche quanto viene scritto nel manuale *Style Manual* [2] della casa editrice Artech House. In questo testo, predisposto come aiuto agli autori dei testi scientifici che la casa editrice pubblica si afferma esplicitamente che lo stile dalla casa prescrive quanto segue.

1. Non si usa la punteggiatura esterna. In una versione più recente di quel manuale, scaricabile dal sito della Artech House, si afferma addirittura che le equazioni in display non vogliono nessuna punteggiatura.

2. Non si comincia mai un capoverso con una equazione né in linea con il testo né in display.

3. Il periodo o la frase che precede una equazione termina con i due punti se l'ultimo elemento della frase è un nome, un aggettivo, un avverbio, un numero di riferimento anche se fra parentesi o in un inciso; non si usa nessuna punteggiatura se la frase termina con un verbo, un articolo, una congiunzione o una preposizione[5].

Forse queste regole potrebbero essere usate anche per qualsiasi oggetto in display inserito nel corso del discorso, non per gli oggetti flottanti.

[5]Secondo lo scrivente sarebbe meglio usare i due punti anche dopo i verbi. Ma qui si sta descrivendo lo stile compositivo della Artech House, non lo stile che bisognerebbe usare in generale.

14.9 Conclusioni

Non è questo il luogo per ripetere quanto è reperibile sulla documentazione del pacchetto `amsmath`; né è questo il luogo per illustrare altri pacchetti e altre funzionalità che la fantasia degli utenti di LaTeX è riuscita a inventare.

Il lettore sappia però che la storia non è finita qui; egli è invitato a esplorare gli archivi internazionali CTAN e scoprirà una varietà infinita di estensioni per la composizione della matematica che non potrà fare a meno di stupirlo; sappia però discernere quanto è stato escogitato alla buona per risolvere un problema contingente, da quanto è stato programmato professionalmente per l'uso generale.

In particolare legga con attenzione il documento `Mathmode.pdf` facente parte della documentazione standard di TeX Live a partire dal 2010, [71]. In questo documento Herbert Voß ha raccolto una infinità di casi di composizione matematica che richiedono comandi speciali e ne indica moltissimi; è una vera miniera da cui attingere idee e soluzioni a problemi compositivi della matematica che non può venire ignorata. Essendo un documento inizialmente scritto da un tedesco per i suoi connazionali, poi tradotto in inglese, si è anche sicuri di trovare informazioni che siano conformi alle tradizioni tipografiche europee, quasi sempre conformi con le norme ISO.

Capitolo 15

LaTeX: composizione di testi letterari e filologici

La composizione di testi letterari o filologici è forse un argomento troppo specializzato per una guida introduttiva come questa.

Tuttavia dopo aver dedicato due capitoli alla matematica, bisogna trattare la filologia che è una scienza molto esatta e che si avvale di una notazione bidimensionale come la matematica. Nello stesso tempo sembra necessario trattare delle classi che consentono di comporre testi letterari con stili assai diversi da quelli dei testi tecnico-scientifici. Sì, è vero che LaTeX si è sviluppato inizialmente nel mondo delle materie tecnico-scientifiche, ma la sua forza compositiva è stata riconosciuta anche da coloro che si occupano di scienze umane e non è un caso che fra i primi umanisti ci siano stati proprio i filologi.

Talvolta è difficile distinguere un testo letterario da un testo filologico perché il testo ha entrambe le caratteristiche; quindi la suddivisione della descrizione che ora si farà con sezioni distinte è puramente semplificativa e una sezione non esclude l'altra.

15.1 La composizione di testi letterari

In un certo senso la composizione di testi letterari, come romanzi, saggi e simili è più semplice di quelli tecnico-scientifici; in realtà segue delle norme e convenzioni di stile compositivo un po' differenti grazie al fatto che la mancanza di numerosi inserti in display permette di essere più rigidi per quel che riguarda la scelta dei font, la giustezza e l'intera geometria della pagina; questa rigidezza si riflette specialmente sullo scartamento che deve assolutamente essere costante e replicarsi esattamente sul recto e sul verso dello stesso foglio.

Se non occorrono altre strutture compositive di sezionamento gerarchico la geometria della pagina è piuttosto semplice e permette di usare le proporzioni della gabbia descritte nei capitoli 6 e 20. Tuttavia merita citare qui alcune

Formato	base per altezza	*Formato*	base per altezza
foolscalp	108 mm × 171 mm	*crown*	127 mm × 191 mm
post	122 mm × 194 mm	*largepost*	137 mm × 210 mm
demi	143 mm × 222 mm	*medium*	146 mm × 229 mm
royal	159 mm × 254 mm	*superroyal*	171 mm × 267 mm
imperial	191 mm × 279 mm		

Tabella 15.1: Formati britannici della pagina

semplici classi preconfezionate che consentono di comporre testi letterari molto semplici.

Fra i pacchetti che offrono queste possibilità non può essere trascurata la classe `octavo`. Questa semplice classe si basa sulla classe `book` ma ha dei formati di carta dai nomi molto britannici (vedi la tabella 15.1) ed è prevista per la composizione in ottavo, proprio come dice il nome. I margini sono prefissati e la gabbia è proporzionata al corpo normale. Sono disponibili tutte le strutture compositive della classe `book` ma, secondo lo scrivente, questa classe è particolarmente indicata per i testi letterari. Si richiama, anzi, l'attenzione sul formato *largepost* la cui altezza corrisponde al formato A5, mentre la larghezza è un poco inferiore; facendo un po' di attenzione e sfruttando adeguatamente le indicazioni della documentazione, è possibile stampare nell'ordine giusto, fronte retro, quattro fogli A4 per poter disporre di un fascicoletto di 8 pagine e 16 facciate che può costituire una piccola brochure già da sola, oppure può costituire una segnatura in ottavo di un documento più vasto. Stampate tutte la segnature anche su una normale stampante per formato A4, le segnature possono essere cucite in modo da formare un libretto completo che risulta di un formato un po' più stretto del formato A5, quando si siano rifilati i margini esterni con una taglierina così da rispettare le dimensioni secondo il disegno originario del formato *largepost*.

Per avere disponibile anche il formato A5 non è difficile modificare la classe (dopo averle cambiato il nome, per esempio in `ottavo`) aggiungendo le tre righe relative a questo formato:

```
\@mainmattertrue
%%%% Opzione a5paper prima dei formati inglesi
\DeclareOption{a5paper}
    {\setlength\paperheight{210mm}%
    \setlength\paperwidth{148.5mm}}
%%%% Seguono le altre opzioni per i formati inglesi
```

Anche la classe `layaureo`, che è stata descritta meglio nel capitolo 6 funziona davvero bene per la stampa di lavori letterari in formato A4 ma con la gabbia e i margini disposti rispettando in parte la proporzione aurea. Naturalmente questa classe serve anche per i documenti tecnico-scientifici, ma, sempre secondo lo scrivente, è molto indicata per i testi letterari.

Nel capitolo 6 viene descritta anche la classe *memoir*; sempre secondo lo scrivente, questa classe è la più versatile, specialmente dopo le aggiunte presenti nella versione del 2008; usando bene le opzioni disponibili e i comandi di personalizzazione è abbastanza agevole disporre di una gabbia adatta al testo letterario, con una scelta assolutamente libera per i font dei titolini correnti in modo da rispettare facilmente le indicazioni fornite nel testo [58], indicate principalmente per i testi letterari.

Nel capitolo 6 è stata descritta anche la classe *suftesi*, esplicitamente creata da un umanista per testi umanistici, anche se non è fuori luogo il suo uso per testi tecnico-scientifici. Il suo pregio è di essere fortemente personalizzabile nei formati e nei layout di pagina; è usabile per comporre riviste complete contenenti molti articoli scritti da autori diversi; il suo creatore la sta già usando per comporre una rivista..

Se però si vogliono scrivere testi letterari di tipo antologico, con le righe numerate sia per le parti in prosa sia per quelle in poesia, allora conviene riferirsi al pacchetto *technica*, il cui nome, almeno nella documentazione, è scritto in greco: τεχνικὰ.

Si tratta di un lavoro molto completo e di largo respiro; consente di comporre il testo letterario puro e semplice, quello relativo ai libretti teatrali, il testo poetico; gli esempi allegati alla documentazione riportano testi in latino, italiano, greco classico, inglese, francese, spagnolo e tedesco, così da mostrare anche le possibilità compositive in lingue diverse dall'italiano. Per ogni esempio è indicato lo stile della casa editrice e/o della collana che è stato riprodotto con una certa (in realtà ottima) approssimazione. La cosa curiosa è che i comandi della classe sono tutti in latino, e l'autore, Gianfranco Boggio-Togna, afferma che l'utente può anche usare i casi per i nomi latini in modo da non urtare la sensibilità dei latinisti, che così possono scrivere dei costrutti di comandi in latino (quasi) perfetto. Il "quasi" è d'obbligo, perché buona parte dei comandi si riferisce a comandi di formattazione che certamente non hanno equivalenti latini, nemmeno cercando sul vocabolario ufficiale edito dalla Tipografia Vaticana[1].

Quello che può trovare un poco stupiti è che gli esempi greci non fanno uso dei font di default di *babel*, e nemmeno del nome *greek* per l'opzione per la lingua greca; l'autore non dice che il suo pacchetto non sia adatto ad essere usato con i font di default e con la normale opzione *greek*, ma lui usa sempre esempi con la lingua *ibycus* che a sua volta fa riferimento ad un altro font greco tramite una codifica chiamata *Beta Code*. L'aspetto dei singoli caratteri è molto simile, anche se non proprio identico ai caratteri mostrati nella tabella 18.12, tuttavia la scrittura con una tastiera latina prevede una corrispondenza dei tasti leggermente diversa e, quel che più conta, i segni diacritici sono tutti indicati invariabilmente *dopo* la lettera a cui si riferiscono (tranne per le maiuscole, dove vanno indicati prima) e vengono ottenuti con tasti diversi da quelli che vengono usati con i font

[1] La lingua latina è quella ufficiale della Santa Sede, da non confondere con lo Stato della Città del Vaticano, la cui lingua ufficiale è l'italiano; vi è anche una Commissione apposita per adattare il latino moderno con neologismi o locuzioni inedite necessarie per riferirsi a oggetti e/o a situazioni della vita contemporanea.

ΕΠΤΑ ΕΠΙ ΘΗΒΑΣ

ΕΤΕΟΚΛΗΣ

Κάδμου πολῖται, χρὴ λέγειν τὰ καίρια
ὅστις φυλάσσει πρᾶγος ἐν πρύμνῃ πόλεως
οἴακα νωμῶν, βλέφαρα μὴ κοιμῶν ὕπνῳ.
εἰ μὲν γὰρ εὖ πράξαιμεν, αἰτία θεοῦ·
εἰ δ' αὖθ', ὃ μὴ γένοιτο, συμφορὰ τύχοι,
Ἐτεοκλέης ἂν εἷς πολὺς κατὰ πτόλιν
ὑμνοῖθ' ὑπ' ἀστῶν φροιμίοις πολυρρόθοις
οἰμώγμασὶν θ', ὧν Ζεὺς ἀλεξητήριος
ἐπώνυμος γένοιτο Καδμείων πόλει.
ὑμᾶς δὲ χρὴ νῦν, καὶ τὸν ἐλλείποντ' ἔτι
ἥβης ἀκμαίας καὶ τὸν ἔξηβον χρόνῳ,
ὥραν τ' ἔχονθ' ἕκαστον ὥστε συμπρεπές,
βλαστημὸν ἀλδαίνοντα σώματος πολύν,
πόλει τ' ἀρήγειν καὶ θεῶν ἐγχωρίων
βωμοῖσι, τιμὰς μὴ 'ξαλειφθῆναί ποτε.

Figura 15.1: Breve estratto dell'opera di Eschilo *Sette contro Tebe*

di default. Questo dipende dal fatto che gli ellenisti molto spesso fanno riferimento al Theasurus Linguae Graecae in rete, e questo Thesaurus ha i suoi testi scritti in Beta Code; per il letterato è quindi molto agevole servirsi del metodo "copia e incolla" per riportare brani già presenti sul Thesaurus e per i quali sia concesso di operare in questo modo. Il letterato risparmia non poco tempo e fatica, ma principalmente risparmia un gran numero di errori di battitura.

Inoltre è curioso che il Beta Code preveda nel file sorgente solo simboli ASCII (e fin qui va bene) ma tutte le lettere devono essere maiuscole nel file sorgente; gli applicativi che consentono di leggere i file sorgente già resi in lettere greche, rendono tutte le lettere in minuscolo, a meno che la maiuscola non sia marcata con un asterisco. La codifica Beta Code di ibycus è molto più semplice, nel senso che le minuscole vengono scritte con lettere minuscole e le maiuscole vengono scritte direttamente con lettere maiuscole; un file sorgente scritto in lettere latine sfruttando il Beta Code non appare a prima vista molto diverso dallo stesso testo scritto in lettere latine con la codifica di default del font usato da `babel`; tuttavia le due codifiche sono profondamente diverse e incompatibili l'una con l'altra.

Perché allora il font greco di default non è stato fatto secondo la codifica Beta Code? Ci sono almeno tre spiegazioni: (*a*) i font CBgreek sono stati fatti estendendo un font (già presente negli archivi CTAN) che aveva già una sua codifica e questa codifica è stata mantenuta; (*b*) la codifica del font preesistente

era stata definita da Sylvio Levy, uno scienziato di origine greca – nato in Brasile e ora operante negli USA – che sapeva usare bene sia la tastiera greca sia quella latina; se un greco aveva scelto quella codifica, aveva le sue buone ragioni, ed era certo che Beccari non poteva arrogarsi il diritto di mettere a disposizione di un pubblico, comprendente anche tutti i greci, un font incompatibile con le abitudini di diteggiatura sulla tastiera greca, dove i diacritici hanno sempre la posizione *prefissa* invece che *postfissa* come nel Beta Code; (*c*) in rete è disponibile il pacchetto `bgreek` che permette di usare i font di default con la notazione del file sorgente che segue le convenzioni Beta Code.

Ciò premesso il pacchetto `technica` va usato con la classe `book`, anche se ne modifica tutte le impostazioni e aggiunge un numero enorme di estensioni; non permette ancora di comporre una edizione critica, ma ci va molto vicino.

Nella figura 15.1 è rappresentato un breve brano in versi della tragedia di Eschilo *Sette contro Tebe*; il parlante, Eteocle, è riportato sul margine sinistro in maiuscolo inclinato, mentre i versi sono numerati con una numerazione che ad un certo punto presenta i versi 12 e 13 scambiati rispetto al codice originale. Si noti che l'esempio della figura 15.1 è stato ricomposto con l'opzione *greek* di `babel`, specificando il modificatore *polutoniko*, tanto per verificare se i font di default, con la loro notazione dei diacritici "prefissa", invece che "postfissa", dessero qualche fastidio al pacchetto `technica`, ma non si sono osservati difetti di sorta.

15.2 Scrivere in greco

Vale la pena, a questo punto, aggiungere poche righe per ricordare come si fa a comporre in greco usando la tastiera latina. Innanzi tutto il file sorgente deve invocare `babel` indicando (solo o) anche la lingua greca; se si vuole scrivere in greco moderno con l'ortografia monotonica, non occorre altro; se però si vuole scrivere in greco classico, o in greco moderno con l'ortografia politonica, bisogna aggiungere una dichiarazione di attributo per la lingua greca, così:

Solo pdflatex

```
\usepackage[...,greek.polutoniko,...,italian]{babel}
```

oppure:

```
\usepackage[...,greek.ancient,...,italian]{babel}
```

Volendo si potrebbe caricare anche il pacchetto `teubner`,

```
\usepackage[...,greek,...,italian]{babel}
\usepackage{teubner}
```

ma di solito, se non occorrono speciali strutture filologiche, l'uso del pacchetto `teubner` è eccessivo.

Le corrispondenze fra i caratteri della tastiera 'latina' con le lettere greche sono riportate nella tabella 15.2. Gli accenti e gli spiriti vanno preposti in una

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	w	x	y	z
α	β	ς	δ	ε	φ	γ	η	ι	ϑ	κ	λ	μ	ν	ο	π	χ	ρ	σ	τ	υ	ω	ξ	ψ	ζ
			>	<	’	‘	~	”	”␣	|			:	;	?			((	))					
			̓	̔	́	̀	̃	̈	’	̣			:	·	;			«	»						

Tabella 15.2: Corrispondenze fra i caratteri della tastiera latina e i segni dell’alfabeto greco

sequenza qualunque alle vocali (o al ρ) *anche* senza l’uso di backslash; lo iota sottoscritto va inserito invece posponendo il segno | alla vocale che lo deve accogliere. Il segno " seguito da uno spazio si traduce in un apostrofo, mentre se precede un iota o un upsilon vi colloca la dieresi; in questa funzione la dieresi può essere anche preceduta o seguita dal segno di accento grave, o acuto o circonflesso. L’unica difficoltà che si incontra a scrivere con la tastiera italiana è che da questa manca una combinazione di tasti per generare l’accento grave e l’accento tilde che funziona da circonflesso per il greco; la tastiera italiana del Mac consente di ottenere questi segni con l’aiuto del tasto ‘comando’, ma con Windows è necessario ricorrere al tasto Alt e alla digitazione dei numeri 96 e 126 rispettivamente sul tastierino numerico; su un laptop Windows è una vera contorsione, ma se lo shell editor lo consente, questi segni possono essere ottenuti con un semplice click del mouse in una apposita tabellina proposta dall’editor. Da Win8 in poi è possibile disporre ud una visualizzazione sullo schermo del layout della tastiera ed è possibile installare diversi driver di tastiera; quindi selezionando il driver per il greco ed avendo sullo schermo la tastiera virtuale, non è diffcile scrivere direttamente in greco con tutti e caratteri possibili e con tutti i diacritici possibili. Sul Mac la tastiera virtuale è disponibile da anni.

Scrivere in greco diventa ora abbastanza facile, anche se sullo schermo appare la versione traslitterata in lettere latine. Si prenda ad esempio il piccolo file seguente:

```
\documentclass{book}
\usepackage[greek,italian]{babel}
\usepackage{teubner}
%
\begin{document}
%
\begin{quote}\begin{otherlanguage*}{greek}\itshape
To’utou q’arin >ap’elip’on se >en Kr’hth|, <’ina t‘a le’iponta
>epidiort’wsh| ka‘i katast’hsh|s kat‘a p’olin presbit’erous, <ws
>eg’w soi dietax’amen, e>’i t’is >estin >an’egklhtos, mi~as
gunaik‘os >an’hr, t’ekna >’eqwn pist’a, m‘h >en kathgor’ia|
>aswt’ias >‘h >anup’otakta. de~i g‘ar t‘on >ep’iskopon
>an’egklhton e>~inai <ws Jeo~u o>ikon’omon, m‘h a>uj’adh,
m‘h  >org’ilon, m‘h p’aroinon, m‘h pl’hkthn, m‘h a>isqrokerd~h,
>all‘a fil’oxenon, fil’agajon. s’wfrona, d’ikaion, <’osion,
```

Τούτου χάριν ἀπέλιπόν σε ἐν Κρήτῃ, ἵνα τὰ λείποντα ἐπιδιορτώσῃ καὶ καταστήσῃς κατὰ πόλιν πρεσβιτέρους, ὡς ἐγώ σοι διεταξάμεν, εἴ τίς ἐστιν ἀνέγκλητος, μιᾶς γυναικὸς ἀνήρ, τέκνα ἔχων πιστά, μὴ ἐν κατηγορίᾳ ἀσωτίας ἢ ἀνυπότακτα. δεῖ γὰρ τὸν ἐπίσκοπον ἀνέγκλητον εἶναι ὡς Θεοῦ οἰκονόμον, μὴ αὐθάδη, μὴ ὀργίλον, μὴ πάροινον, μὴ πλήκτην, μὴ αἰσχροκερδῆ, ἀλλὰ φιλόξενον, φιλάγαθον. σώφρονα, δίκαιον, ὅσιον, ἐνκρατῆ, ἀντεχόμενον τοῦ κατὰ τὴν διδαχὴν πιστοῦ λόγου, ἵνα δυνατὸς ᾖ καὶ παρακαλεῖν ἐν τῇ διδασκαλίᾳ τῇ ὑγιαινούσῃ καὶ τοὺς ἀντιλέγοντας ἐλέγκειν. T. 1.5-9

Figura 15.2: Esempio di composizione in greco

```
>enkrat~h, >anteq'omenon to~u kat'a t'hn didaq'hn pisto~u l'ogou,
<'ina dunat'os <~h| ka'i parakale~in >en t~h| didaskal'ia| t~h|
<ugiaino'ush| ka'i to'us >antil'egontas >el'egkein.
\null\hfill{\upshape{T. 1.5-9}}
\end{otherlanguage*}
\end{quote}
%
\end{document}
```

Solo pdflatex

Dopo averlo elaborato con pdflatex, si ottiene il risultato mostrato nella figura 15.2 (dove fra l'altro il testo è composto con il font Lipsiano). Si noti anche che il testo sorgente non usa mai la traslitterazione 'c' per il sigma terminale, ma la lettera 's'; questo è voluto per agevolare chi scrive in greco, ma questa non è la sua lingua madre tanto che ha bisogno di usare la tastiera latina. Scrivendo abitualmente in una lingua occidentale sarebbe troppo facile sbagliarsi nello scrivere il sigma terminale; per questo il font è congegnato in modo da rendersi conto da solo di quando il sigma cade in posizione terminale, e automaticamente esegue una legatura di sostituzione, introducendo un sigma terminale, benché nel file sorgente comparisse una 's'. Questo meccanismo è talmente robusto, che diventa difficile scrivere un σ iniziale o mediale isolato; per questo scopo bisogna introdurre nel file sorgente `sv`, dove la 'v' serve per far credere al meccanismo di sostituzione che la 's' introdotta non sia terminale; come si vede dalla tabella 15.2, la lettera latina 'v' non corrisponde a nessun segno greco.

Se si deve scrivere a lungo in greco e si dispone di una tastiera bilingue o, meglio ancora, di mezzi comodi per scrivere direttamente in greco nel file sorgente, si può usare la codifica *utf8*, in questa maniera:

Anche lualatex e xelatex

```
\documentclass{book}
\usepackage{iftex}% Per riconoscere il programma
                  % di composizione
\ifPDFTeX % compilazione con pdflatex
   \usepackage[T1]{fontenc}
   \usepackage{lmodern}
```

```
   \usepackage[utf8]{inputenc}
   \usepackage[greek.ancient,italian]{babel}
\else % compilazione con xelatex o lualatex
   \usepackage{polyglossia}
   \setmainlanguage{italian}
   \setotherlanguage[variant=ancient]{greek}
   \usepackage{fontspec}[Ligatures=TeX]
   \newfontfamily{greekfont}{CMU Classical Serif}
\fi
\begin{document}

\begin{otherlanguage*}{greek}%

   Τούτου χάριν ἀπέλιπόν σε ἐν Κρήτῃ, ἵνα τ‘α λείποντα
   ἐπιδιορτώσῃ κα‘ι καταστήσῃς κατ‘α πόλιν πρεσβιτέρους, ὡς
   ἐγώ σοι διεταξάμεν, εἴ τίς ἐστιν ἀνέγκλητος, μιᾶς
   γυναικ‘ος ἀνήρ, τέκνα ἔχων πιστά, μ‘η ἐν κατηγορίᾳ
   ἀσωτίας ’‘η ἀνυπότακτα. δεῖ γ‘αρ τ‘ον ἐπίσκοπον
   ἀνέγκλητον εἶναι ὡς θεοῦ οἰκονόμον, μ‘η αὐθάδη,
   μ‘η ὀργίλον, μ‘η πάροινον, μ‘η πλήκτην, μ‘η αἰσχροκερδῆ,
   ἀλλ‘α φιλόξενον, φιλάγαθον. σώφρονα, δίκαιον, ὅσιον,
   ἐνκρατῆ, ἀντεχόμενον τοῦ κατ‘α τ‘ην διδαχ‘ην πιστοῦ λόγου,
   ἵνα δυνατ‘ος ᾖ κα‘ι παρακαλεῖν ἐν τῇ διδασκαλίᾳ τῇ
   ὑγιαινούσῃ κα‘ι το‘υς ἀντιλέγοντας ἐλέγκειν.

\null\hfill{\upshape{T. 1.5-9}}
\end{otherlanguage*}

Altro testo in italiano.
\end{document}
```

Nel precedente esempio di codice sorgente, si è mostrato come caricare il file di estensione *iftex*, che mette a disposizione dell'utente i tre test condizionali `\ifPDFTeX`, `\ifLuaTeX` e `\ifXeTeX` con i quali si può controllare con quale programma di compilazione si sta lavorando; Se si sta lavorando con pdflatex vengono eseguiti i comandi di impostazione del preambolo per l'uso con questo compilatore; altrimenti si suppone che l'utente non si sbagli e compili o con xelatex o con lualatex (non sarebbe difficile eseguire i test necessari per verificarlo e per emettere un messaggio di errore nel caso che si usi un altro programma di compilazione) e si imposta il preambolo per lavorare con uno di questi due programmi mediante l'uso di font OpenType. Non si sono specificati i font latini, che per impostazione predefinita sono i Latin Modern. Il resto, in particolare il corpo del documento è lo stesso in entrambi i casi.

15.3 Font adatti ai testi classici

Fra i font usabili con il sistema TEX (in numero sterminato) pochi in verità contengono le legature presenti nei testi classici a stampa, secondo la tradizione dei secoli XVI e XVII. Questi font contenevano sia la s 'lunga' sia la s 'corta', la prima usata come s iniziale e mediale, la seconda come s finale e come secondo elemento della doppia s: *ss* ⟶ *ß*. Contenevano inoltre diverse legature per esempio nei nessi 'ct' e 'st'; contenevano anche le legature æ e œ, assenti nel latino classico, ma entrate in uso sia nei codici tardomedievali, sia negli incunaboli e nelle cinquecentine, assieme all'uso della 'u' minuscola per indicare sia la 'u' sia la 'v' e 'V' maiuscola per indicare sia la 'U' sia la 'V'. La maggior parte di queste abitudini tipografiche cessarono verso la fine del Settecento a partire dai Paesi Bassi per poi estendersi al resto d'Europa e nel resto del mondo.

Desiderando comporre testi classici o antichi conservando l'atmosfera della composizione originale, si possono usare font OpenType 'esperti' che contengano anche le numerose legature. A questo proposito conviene segnalare il lavoro di Massimiliano Dominici [18] che ha usato i font FellType per i quali ha dovuto creare l'insieme di file di descrizione di questi font, rimodificandoli in modo da poterli usare con pdflatex, che non gestisce i font OpenType direttamente; anche i TrueType usati da Dominici si estendevano ben oltre i 256 caratteri gestibili con LATEX.

Per lavorare con pdflatex possiamo fare le seguenti considerazioni. Solo pdflatex

Probabilmente ne esistono altri, ma qui si segnalano i font chiamati con la sigla Kp-fonts[2] e richiamabili nei file da comporre con pdflatex mediante il pacchetto `kpfonts`. Secondo la descrizione che ne ha fatto l'autore Christophe Caingnaert [12], sono direttamente derivati dal font Palatino, adeguatamente reinterpretati dando loro un asse ottico distintamente obliquo. La proprietà di questi font è che dispongono delle serie chiara, media e nera; il maiuscoletto vero può essere composto con lettere 'minuscole' di dimensioni piccole o medie; il maiuscoletto chiaro, medio e nero può essere composto anche in forma inclinata; esistono le varianti che contengono oppure escludono le legature 'ct' e 'st' e usano di default la 'Q' a coda lunga oppure la 'Q' normale; nello stile *veryoldstyle* possono usare anche altre legature; sono presenti sia la s lunga sia la s corta; sono componibili sia con la codifica *OT1* sia con quella *T1*; dispongono delle cifre 'minuscole'; consentono la composizione della matematica in vari stili moderni e antichi, utili anche per le edizioni critiche di testi matematici antichi. Le possibili varianti si ottengono con opportune opzioni esplicitate durante la chiamata del pacchetto `kpfonts`. I simboli disponibili sono moltissimi, molti di più che non unendo le collezioni dei simboli matematici di LATEX e quelli della American Mathematical Society. Questi font sono quindi raccomandabili sia per i testi letterari, sia per le edizioni critiche, sia per testi moderni, anche se contengono molta matematica, composti con una collezione di font molto completa, forse una delle più ricche fra quelle distribuite con il sistema TEX.

[2] Forse le lettere 'kp' derivano dal nome di Kepler, essendo stati sviluppati per il JOHANNES KEPLER project. Non hanno invece nessuna relazione con i font Kepler della Adobe.

Nella figura 18.4 nella pagina 399 si vede l'uso di questi font sia con lo stile compositivo *oldstyle*, sia con lo stile *veryoldstyle* specificati come opzioni al pacchetto *kpfonts*.

lualatex o xelatex

Per lavorare con lualatex e xelatex bisognerebbe esaminare le particolarità dei font OpenType disponibili sulla propria macchina; sul mio Mac tra quelli di sistema, quelli aggiunti dal sistema TEX e quelli che corredano alcuni software, ne ho alcune centinaia; difficili da contare perché i nomi delle famiglie sono diverse centinaia, ma le famiglie contengono da da uno a una dozzina di file di font, i quali a loro volta prevedono diverse varianti. Certamente c'è solo l'imbarazzo della scelta; diversi font contengono varianti che prevedono le legature antiche. Si suggerisce di usare il programma otfinfo da lanciare dalla finestra comandi con le debite opzioni per sapere di quali varianti dispone un dato font; per quali lingue è adatto, se ha corpi ottici, eccetera. Siccome bisogna passare a questo programma il nome completo di path per riceverne le informazioni richieste, bisogna sapere dove si trovano nel disco i font OpenType; ovviamente ogni sistema operativo ha le sue cartelle, quindi qui non si può dire altro se non che i font OpenType distribuiti con il sistema TEX si trovano nella cartella `.../texmf-dist/fonts/opentype/`.

Per gli altri font chi scrive può dirlo solo per il Mac: essi si trovano nella cartella `~/Library/Fonts` (anche quelli installati fra quelli del sistema TEX) e le informazioni richieste si ottengono cliccando col destro, oppure control + click, il nome di un font `.otf` qualsiasi in quella cartella, che apre l'applicativo FontBook.app; la finestra di questo applicativo contiene nella parte di sinistra l'elenco alfabetico di tutti i font installati sulla macchina, e di ciascuno può mostrare le informazioni scritte, un esempio di testo composto con quei font, la tabella dei caratteri, eccetera.

15.4 La composizione del tedesco classico

Fra le lingue classiche e moderne solo il tedesco è molto noto per usare i font gotici, decisamente diversi da quelli latini, anche se le lettere, semanticamente sono le stesse.

Dalla prima Bibbia di Gutenberg alla fine della seconda guerra mondiale il gotico è stato spesso usato ma, anche impropriamente, come simbolo di una nazione. Per il nazismo era stato l'alfabeto caratterizzante quel regime.

I modelli della scrittura a cui in italiano ci si riferisce con l'aggettivo *gotico* sono sostanzialmente tre: (*a*) il textur usato da Gutenberg, (*b*) il fraktur, usato nei secoli più recenti e, (*c*) lo schwabacher usato come alfabeto elegante, sempre di tipo gotico, ma con le lettere più rotondeggianti, specialmente le maiuscole. Nella figura 15.3 sono mostrati alcuni alfabeti gotici disponibili con il sistema TEX.

In realtà anticamente non c'era nessuna distinzione fra la 'i' e la 'j'; nei font mostrati nella figura 15.3 le due lettere minuscole sono leggermente differenziate, mentre le maiuscole sono rimaste identiche.

Font textur

abcdefghijklmnopqrstuvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZ

Font fraktur

abcdefghijklmnopqrstuvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZ

Font schwabacher

abcdefghijklmnopqrstuvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZ

Capilettera

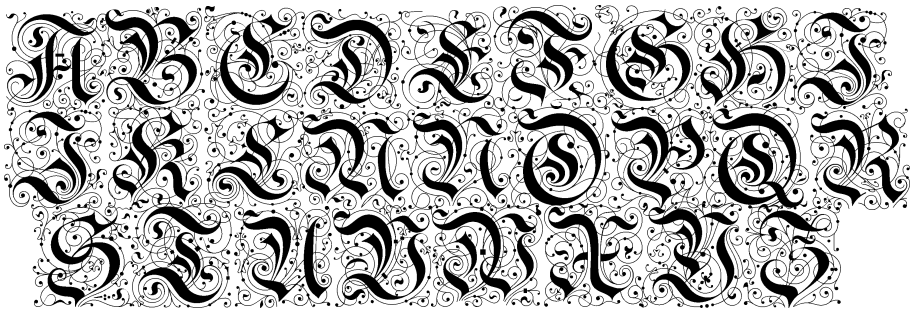


Figura 15.3: Alcuni alfabeti gotici che includono anche i capilettera ornati

Si noti che nelle distribuzioni di TEX Live recenti, certamente nella distribuzione del 2010, i tre font textur, fraktur e schwabacher sono disponibili come font vettoriali e sono già disponibili come tali senza bisogno di ricorrere all'installazione delle loro mappe. I font corrispondenti ai bellissimi e ornatissimi capilettera sono in forma bitmapped; forse la loro delicata e fitta ornamentazione impedisce loro di venire trasformati in font vettoriali, ma nelle prove fatte su carta non si nota nessun difetto tipico dei font bitmapped; anche sullo schermo si presentano molto bene.

Per fare uso di questi font basta impiegare il pacchetto *mfnfss*, che definisce i comandi usuali `\gothfamily` e `\textgoth` per selezionare il font textur, `\frakfamily` e `\textfrak` per selezionare il font fraktur, `\swabfamily` e `\textswab` per selezionare il font schwabacher. Per selezionare il font dei capilettera il file di documentazione suggerisce una macro che contemporaneamente

Solo pdflatex

sceglie il font e lo compone ribassato; qui la si riporta adeguatamente "corretta" visto che come è riportata nella documentazione fa riferimento ad una macro non descritta:

```
\newcommand*\yinitial[1]{%
 \hangindent=2.54cm
 \hangafter=-4
 \hskip-3.24cm
 \lower-2.7mm
 \hbox{\fontencoding{U}\fontfamily{yinit}\selectfont #1}%
 \hskip1.5mm}
```

dove le misure indicate possono essere ritoccate a discrezione dell'utente.

I caratteri contengono le necessarie legature antiche e per le umlaut consentono di usare sia il doppio apice (`"a` produce ä) sia l'asterisco (`*a` produce una 'a' a cui è sovrapposta come una segno diacritico una minuscola 'e'); queste seconde legature sono specificamente descritte per lo schwabacher.

In sostanza, se si dovesse riprodurre una pagina della Bibbia di Gutenberg, lo si potrebbe fare in modo quasi identico, compresi i capilettera, anche se questi hanno un disegno di un paio di secoli successivi a quelli usati da Gutenberg stesso.

15.5 La composizione di testi in lingue classiche

In questo breve paragrafo si accenna alla scrittura in latino e in greco, che si ritiene che siano le lingue classiche più diffuse fra gli umanisti italiani. Non si scende nei dettagli di che cosa sia necessario fare con scritture molto particolari; si segnala solo che fra i font del sistema TEX compare anche il Lineare A, per esempio, al fine di riportare testi molto antichi.

15.5.1 Scrivere in latino

Solo pdflatex

Scrivere in latino non richiede altro che specificare l'opzione *latin* a `babel`; tuttavia se si desidera estendere la funzionalità alla marcatura della lunghezza delle sillabe, bisogna provvedere alla specificazione dell'attributo della lingua:

```
...
\usepackage[...,latin,...]{babel}
\languageattribute{latin}{withprosodicmarks}
...
```

Un altro attributo che si può specificare è quello relativo all'ortografia medievale:

```
...
\usepackage[...,latin,...]{babel}
\languageattribute{latin}{medieval}
...
```

oppure al latino classico:

```
...
\usepackage[...,latin,...]{babel}
\languageattribute{latin}{classic}
...
```

La differenza nel latino medievale è che in lettere minuscole le lettere 'u' e 'v' si scrivono entrambe 'u', mentre in lettere maiuscole esse vengono scritte entrambe 'V'. Di per sé non sarebbe un problema per il tastierista a scriverle come appena detto, ma quando ci sono delle trasformazioni automatiche in minuscolo o in maiuscolo, i comandi `\MakeUppercase` e `\MakeLowercase` devono operare correttamente anche con l'ortografia medievale. Questa regola vale anche per il latino classico, ma questo non usa le legature æ ed œ, mentre il latino medievale ne fa ampio uso. Le due ortografie differiscono anche per la divisione in sillabe.

Una cosa che va osservata attentamente è che la divisione in sillabe sia con l'ortografia moderna sia con quella medievale funziona bene in modo fonetico; quella del latino classico funziona con le regole etimologiche tradizionalmente usate dai latinisti; buona parte dei suffissi e dei prefissi sono riconosciuti correttamente in modo da dividerli etimologicamente, ma l'insieme di pattern per la sillabazione latina non garantisce sempre una divisione corretta. Per questo diventa particolarmente importante l'uso del carattere attivo `"` che introduce un punto di cesura facoltativo senza impedire la divisione del resto della parola. Si scriverà pertanto `transi"erat` per consentire la divisione dopo la 'i' senza impedirla negli altri punti leciti.

Le date vengono composte con il calendario gregoriano, ma i numeri relativi ai giorni e agli anni vengono scritti con la numerazione romana sottrattiva; in maiuscoletto il numero dei giorni, in maiuscolo quello degli anni. Un tentativo di trasformare la data gregoriana in una data "ab urbe condita" non ha avuto seguito (per ora).

Per usare gli altri programmi di compilazione non ci sono problemi con i font latini; per la scrittura medievale e liturgica sono sempre disponibili i caratteri æ ed œ, semplici e accentati; veramente œ́ non è codificato UNICODE, quindi o lo si ottiene con la macro per l'accento acuto, o si usa il carattere UNICODE `U+0301`, *self combining acute accent* che va indicato in posizione postfissa, `œ^^^^0301`, con una scrittura veramente inconsueta. Certo se ne può costruire una macro, ma il problema non è quello. Il vero problema è che per polyglossia il team *tex-hyphen* prende tempo nell'accettare ciò che i curatori della sillabazione latina propongono e, in particolare, per lualatex, che carica i pattern al momento del bisogno, non accetta la definizione di nuovi nomi di lingue e di conseguenza non accetta nuovi file `gloss-`⟨*latin*⟩ con nomi diversi per le varietà delle ortografie latine. Il tutto ora (inizio 2017) è in mano a nuovi curatori e speriamo che questo intoppo si sblocchi in breve tempo. Dal 2018 è possibile usare la selezione del latino classico, moderno e medievale, e liturgico con nomi espliciti, quindi la funzionalità di LuaLaTeX è sfruttata al meglio.

xelatex o lualatex

15.5.2 Scrivere in greco

Si è già detto molto sia in questo capitolo, sia nel paragrafo 15.2. Non si ripeteranno quelle cose, qui, ma si sottolinea l'uso opportuno e consigliabile
Solo pdflatex del pacchetto *teubner* nella versione 4.x. Infatti in questa versione sono stati aumentati molto i comandi per gestire le varie forme dei font greci (CBgreek) tenendo conto che in un testo letterario o filologico verosimilmente si passa attraverso un'alternanza di testo in lingua greca con altro testo in una lingua occidentale.

La gestione dei segni diacritici è anche molto migliorata e permette di evitare di usare le decine di macro strane per comporre i caratteri marcati con i loro diacritici senza ricorrere al meccanismo delle legature; questo si rende necessario specialmente quando si usa il font Lipsiano i cui segni hanno disegni dalle forme delicate che richiedono crenature accurate.

Quando il testo greco è abbondante, sarebbe desiderabile che l'umanista disponesse di una tastiera bilingue, oppure di comodi mezzi software per disporre di una tastiera virtuale su cui comporre il testo greco. In queste condizioni il 'tastierista' può comporre direttamente in greco senza doversi ricordare nessuna corrispondenza fra la tastiera con caratteri latini e quella con i caratteri greci. Però necessita dell'opzione *utf8* per la codifica di entrata; questa particolare opzione permette di tradurre i segni greci UNICODE, che appaiono sullo schermo quando si compone il file sorgente, nei corrispondenti comandi interni (LICR, LaTeX Internal Character Representation) che pdflatex usa e che solo in fase di scrittura del file di uscita trasforma nei caratteri presenti poi per il font specificato con la codifica di uscita; si veda anche il capitolo 26.

Per il greco esistono almeno tre metodi di divisione in sillabe, corrispondenti alle tre ortografie o varietà della lingua: la sillabazione per l'ortografia monotonica, quella per l'ortografia politonica moderna e quella per l'ortografia politonica antica. Si tenga presente che di default le impostazioni per la sillabazione greca con qualunque ortografia prevede una lunghezza minima della prima e dell'ultima sillaba di qualunque parola formata da una sola lettera, quando in italiano esse devono essere formate da almeno due lettere, e nelle altre lingue occidentali con molte parole che finiscono in consonante la prima sillaba deve essere di almeno due lettere e l'ultima sillaba di almeno tre lettere. Queste vistose differenze fanno sì che la prosa normale possa avere capoversi che terminano con un righino contenente una sola lettera, tipograficamente molto brutto; LaTeX pone una fortissima penalità alla cesura nella penultima riga di un capoverso, ma questo non impedisce che possa verificarsi quell'evento.

I vari esempi di testo greco classico scritti in questa guida sono stati tutti composti usando il pacchetto *teubner* e l'attributo `ancient` e senza avvalersi di una tastiera greca né della codifica *utf8*. A questo proposito, visto che oggi gli elaboratori con schermo *touch screen* diventano sempre più diffusi, disponendo di un tale dispositivo, non è impossibile scrivere direttamente in greco toccando lo schermo in corrispondenza dei tasti virtuali della tastiera greca; basta aver installato il driver della tastiera greca ed aver toccato la sua icona nella barra

opportuna dello schermo. Ecco allora che i sistemi *touch screen*, che lo scrivente considerava validi solo per certi usi che nulla hanno a che far con la composizione dei testi, diventano utili proprio a questo scopo quando si debbano usare alfabeti diversi o disposizioni diverse dei tasti.

Solo xelatex e lialatex

Va poi sottolineato che per scrivere agevolmente con alfabeti diversi da quello latino è utile (o necessaria) una tastiera apposita, come detto sopra, ma diventa utilissimo anche usare il programma di composizione xelatex o lualatex; come già spiegato, questi programmi consentono di usare la codifica UNICODE e di gestire direttamente i font OpenType, cosicché la scelta dei font latini, greci, cirillici, eccetera, non è più legata a quanto offre di default il sistema TEX, ma si estende alla moltitudine di font che formano la dotazione del sistema operativo della macchina che si sta usando. Esso apre le porte all'impiego di qualsiasi font commerciale, gratuito o a pagamento, senza bisogno di ricorrere alle procedure di installazione necessarie con pdfLATEX, che però non può gestire direttamente i font OpentType. Per il resto i programmi di composizione xelatex e lualatex hanno un mark-up che differisce da quello di LATEX solo per qualche dettaglio riguardante la gestione dei font, ma per il resto un file sorgente, concepito per essere elaborato con pdfLATEX, spessissimo viene compilato senza modifiche anche da XƎLATEX e LuaLATEX.

Vale la pena di citare il pregevolissimo lavoro fatto da Antonis Tsolomitis che ha reso disponibili i font greci predisposti dalla società GFS (Greek Font Society) in formato OpenType e che questa società ha distribuito in uso gratuito al pubblico; Tsolomitis ha reso questi font accessibili anche a pdflatex; creando i debiti file virtuali e codificando questi font con la codifica LGR, come i font greci preinstallati. I file per usarli sono i seguenti:

`gfsartemisia` corrisponde al pacchetto da invocare per usare i font greci e latini Artemisia.

`gfsbaskerville` corrisponde al pacchetto da invocare per usare i font greci Baskerville, ed è caratterizzato da diverse forme per i glifi iniziali o mediali di gamma, theta, rho e per le legature anticheggianti fra sigma e theta, sigma e chi, upsilon e iota, omicron e sigma e per il doppio tau; si possono usare altre legature, riguardanti specialmente il dittongo omicron-upsilon, mediante comandi specifici.

`gfsbodoni` corrisponde al pacchetto da invocare per usare i font latini e greci Bodoni caratterizzati da lettere latine dalle grazie finissime e dal forte contrasto e da lettere greche dalle forme leggermente calligrafiche.

`gfscomplutum` corrisponde al pacchetto da invocare per usare i font greci Complutum con forme anticheggianti e difficile da leggere per chi non ha familiarità con questo stile.

`gfsdidot` corrisponde al pacchetto da invocare per usare i font latini e greci Didot; questo è il font che si trova usato più spesso nei testi greci; questa particolare polizza distingue la forma della theta iniziale aperta, dalla

forma finale chiusa; altre legature sono possibili mediante comandi appositi; la forma obliqua è ottenuta automaticamente mediante il font Olga.

`neohellenic` corrisponde al pacchetto da invocare per usare i font greci di tipo bastoncino Neohellenic; il font lineare o bastoncino è caratterizzato dalla mancanza di chiaroscuri e tutti i tratti sono sostanzialmente della stessa larghezza, abbastanza sottile.

`gfsporson` corrisponde al pacchetto da usare per invocare i font inclinati Porson, caratterizzati da un disegno moderno, ma con l'accento circonflesso ridotto a un cappellino rotondeggiante; purtroppo la documentazione non offre nessun esempio d'uso del font Porson, ma basta scrivere un breve testo per verificare lo stile di questo font.

`gfssolomos` corrisponde al pacchetto da usare per invocare i font greci Solomos la cui forma è molto calligrafica con i discendenti delle lettere minuscoli abbastanza ornati da riccioli e curve sinuose.

Scrivendo nel terminale il comando `texdoc` ⟨*pacchetto*⟩ (per esempio `texdoc gfsdidot`) si apre il file di documentazione che nelle ultime pagine mostra un po' di testo, eventualmente contenente un po' di matematica, dove si possono notare le caratteristiche del font; purtroppo questi testi sono quasi tutti scritti in greco monotonico, quindi non è possibile osservare la forma dei segni diacritici.

Non si tenga conto in tutti questi file di documentazione del paragrafo intitolato "Transformations by dvips"; i medesimi effetti si ottengono oggi lavorando direttamente con pdflatex probabilmente anche negli anni in cui Tsolomitis creava queste funzionalità aggiuntive.

Il lavoro di Tsolomitis è decisamente importante perché integra molto bene sia il testo latino, sia quello greco, fra testo e matematica. La scelta di integrare con alcuni font i segni latini insieme a quelli greci permette di eseguire degli abbinamenti ottimali, che sono poi quelli che generalmente fanno i tipografi greci in Grecia.

Si noti ancora che Antonis Tsolomitis ha prodotto anche un file `txfontsx` da usare insieme al pacchetto per i caratteri latini `txfonts`. Le due serie di caratteri sono perfettamente integrate. Ne tengano conto coloro che debbono comporre in greco, certamente monotonico, ma anche in generale in greco politonico, integrando il testo con i corrispondenti caratteri latini. Come parte interessata alla codifica *LGR*, devo però avvisare che con i font di Tsolomitis è difficile sfruttare appieno le funzionalità del pacchetto `teubner`, perché non tutti i glifi della codifica *LGR* sono sono presenti nelle polizze di Tsolomitis. Gli stessi problemi nascono dal confronto sulla dotazione di caratteri dei font diritti rispetto a quelli inclinati, che non si corrispondono completamente. Con i font Times eXtended, il pacchetto `teubner` ha posto rimedio, ma non è detto che quella soluzione possa andare bene anche con gli altri font descritti qui.

Infine Tsolomitis si è dedicato alla creazione di una famiglia di font OpenType che contengono i caratteri latini, greci e cirillici, già presente nella dotazione di

TEX Live; si tratta della collezione NewComputerModern il quale è accompagnato anche dalla collezione dei font matematici; si legga il file di documentazione con `texdoc newcomputermodern` per verificare le famiglie, le serie e le forme insieme a qualche formula matematica.

15.6 La composizione di testi filologici

La pubblicazione di scritti filologici include le edizioni critiche e include scritti in prosa o in versi in lingue scritte non necessariamente in caratteri latini. Essi sono scritti letterari, ma in più richiedono diversi apparati critici formati da annotazioni composte in varie maniere. Questi apparati di note spesso compaiono assieme nella stessa pagina, ma devono venire distinti in modo inequivocabile gli uni dagli altri e, chiaramente, non basta solo un filetto orizzontale per distinguerli.

Serve anche una numerazione delle singole righe sia per i testi in prosa, sia per quelli in poesia; generalmente i numeri sono nel margine esterno e sono esplicitati quelli multipli di 5; le note, invece che legate a richiami come numerini a esponente, possono essere legate al numero di riga, per cui nel richiamo bisogna ripetere una breve sequenza di testo a cui la nota si riferisce.

Chiaramente LATEX non è in grado da solo di svolgere questi compiti e bisogna ricorrere a pacchetti di estensione.

Il più noto e storico pacchetto di estensione si chiama *ledmac* ed è in grado di comporre ogni aspetto delle edizioni critiche; la sua documentazione si legge con `texdoc ledmac`; oggi è però soppiantato dalla sua versione estesa costituita dal pacchetto *eledmac*; documentazione `texdoc eledmac`; ultimamente il curatore ha anche prodotto *reledmac*, documentazione `texdoc reledmac`.

L'autore di *ledmac*, Peter Wilson, ha cercato di implementare le capacità sviluppate dai pacchetti di macro per Plain TEX: *edmac*, *tabmac* e *edstanza* aggiungendovi del suo. Oggi il curatore è Maïeul Rouquette, che ha esteso ulteriormente le funzionalità del pacchetto originale di Wilson, adattandolo anche a mezzi moderni di programmazione del sistema TEX.

Esso è particolarmente indicato per le edizioni critiche e il suo forte sono le numerazioni marginali e gli apparati di note sia a piè di pagina sia come note finali a fine capitolo o a fine documento. È possibile legare le note finali al numero di pagina e di riga, invece che a richiami numerici ad esponente. Anche per quel che riguarda l'indice analitico è possibile fare riferimento anche al numero di riga invece che solamente alla pagina.

Altri pacchetti per edizioni critiche sono *endnotes* specializzato nelle note finali, e *poemscol* destinato alle forme poetiche, ai testi in versi.

Va per esempio citato il pacchetto di estensione *opcit* che dispone di un suo particolare file di stile per la composizione della bibliografia con BIBTEX; il pacchetto consente di scrivere i riferimenti bibliografici in nota, invece che in elenco a fine capitolo o a fine documento; le citazioni successive alla prima vengono ripetute in forma abbreviata nella forma "Autore, *op. cit.*"; se la precedente citazione corrisponde alla stessa opera, il comando di citazione riporta solamente

"*Ibidem*, p. 33" (dove l'indicazione della pagina, la post-citazione, costituisce l'argomento facoltativo del comando `\cite`). Sono possibili diverse personalizzazioni così che gli umanisti possano continuare ad usare il loro metodo tradizionale di citazione in nota. L'uso di LaTeX e dei suoi pacchetti assicura l'aspetto omogeneo delle citazioni in modo da garantire la massima professionalità compositiva. La documentazione del pacchetto si trova in `.../doc/latex/opcit/opcit.pdf`.

Quando si va nella filologia vera e propria, le cose si possono fare molto dure, anche perché i filologi delle varie discipline hanno notazioni diverse per rendere gli stessi concetti.

Chi scrive ha collaborato con un filologo di greco classico e con un filologo di lingua copta altomedievale; come prevedibile, i due filologi non erano d'accordo su come esprimere i concetti comuni alle due lingue; questa non è una critica, ma una osservazione; anch'io che scrivo usando molta matematica ho dovuto arrangiarmi con i simboli disponibili perché certi concetti che dovevo esprimere, malgrado le norme ISO, non avevano un simbolo specifico, così alla fine ho dovuto usare simboli che in altre branche della matematica hanno significati diversi. Succede in ogni scienza e anche all'interno della stessa scienza.

Per quanto concerne il greco classico, come già detto, esiste il pacchetto *teubner* già contenuto in ogni installazione aggiornata e completa e abbastanza ben documentato mediante il file `.../doc/latex/teubner/teubner-doc.pdf`. Però per ora questo pacchetto funziona solo con pdflatex. Si noti che l'interfaccia fra *babel* e il greco è cambiata nel 2020, per cui ora *teubner*, versione 4.x, funziona solo con una versione aggiornata dell'installazione del sistema TeX.

Dalla versione 3.x il pacchetto *teubner* dispone di una nuova funzionalità, oltre a macro per gli accenti che rendono inutile il ricorso a quel centinaio di macro per accedere per nome ai caratteri accentati invece che con le comode legature, permette di far convivere i font con codifica *LGR* virtualmente con tutti i font disponibili sull'elaboratore che si sta usando. Già la versione 2.9 era compatibile con i font CM-super e i font Latin Modern, tutti con la codifica *T1*. Con la versione 3.x il pacchetto riconosce da solo se si stanno usando i font Times o Palatino eXtended e si crea da solo i file di descrizione dei font necessari per questa convivenza. Il pacchetto mette a disposizione anche il comando `\ifFamily` che crea le necessarie associazioni anche con famiglie di font diverse da quelli sopra menzionati.

Per esempio, se si volessero usare il font Helvetica come font latini proporzionali senza grazie, allora basterebbe dare i seguenti comandi:

```
\renewcommand{\sffamily}{phv} % Font latini senza grazie
\ifFamily{phv}{lmss}          % Font greci senza grazie
```

In questo modo *teubner* genera i file di descrizione dei font che sostituiscono agli inesistenti Helvetica codificati *LGR* i font greci della collezione CBgreek con la codifica giusta; certo i segni grafici sono diversi, ma se è possibile i segni greci sono meno vistosi e ingombranti dei segni lineari con lo stile Helvetica. Nessuno vieta poi di usare lualatex o xelatex, come mostrato alla fine di questo capitolo,

così da evitare le piccole acrobazie necessarie per poter usare il programma pdflatex, che non è in grado di gestire da solo i font codificati in UNICODE.

Il pacchetto consente di accentare ogni lettera con qualsiasi segno diacritico, persino con accenti impilati, collocandoli correttamente tenendo conto dell'inclinazione dei caratteri. Esso può inserire segni speciali sopra o sotto singole lettere o gruppi di lettere; può racchiudere gruppi di lettere con delimitatori speciali per indicare la lectio incerta, o altre simili indicazioni per interpolazioni o omissioni; può numerare i versi di poemi anche con due serie di numerazioni contemporaneamente; può disegnare le metre dei vari versi con una notevole ricchezza di segni che consentono anche di specificare, per esempio, se una sillaba ancipite è più sovente lunga invece che breve, eccetera.

Va detto che Gianfranco Boggio-Togna ha prodotto anche il pacchetto *metre* che svolge alcune funzioni svolte dal pacchetto *teubner*. I due pacchetti differiscono per la loro estensione, ma anche, quando si tratta delle metre, le notazioni ritmiche dei versi, per come sono prodotti i simboli da usare; *teubner* usa un font particolarmente disegnato allo scopo, mentre *metre* usa segni costruiti componendoli con i simboli dei vari font già disponibili, per esempio, usando simboli tratti dalle polizze per la matematica.

Per quanto concerne il copto, oltre ai font necessari per scrivere con i caratteri altomedievali, c'è una certa varietà di simboli e segni che permettono di annotare il testo antico; con le distribuzioni recenti di TeX Live, anche il pacchetto *coptic* è già preinstallato.

Certamente esistono altri pacchetti in rete destinati a risolvere problemi specifici di composizione filologica; tuttavia ogni lavoro di filologia richiede la predisposizione di comandi particolari per problemi particolari; lo scrivente è al corrente che da alcuni anni a Pisa si sta costruendo l'edizione critica delle opere del matematico Maurolico; sono già disponibili diverse parti, ma l'opera non è finita; si capisce bene che, nonostante il Maurolico scrivesse in latino, egli scriveva di matematica e di geometria quando le convenzioni a cui siamo oggi abituati non esistevano ancora, non parliamo delle norme ISO! I curatori di questa edizione critica stanno lavorando con passione ma devono inventarsi soluzioni nuove per problemi di composizione che finora non sono mai stati affrontati, meno che mai risolti.

In Italia, oltre che a Pisa, si è al corrente che anche alla Pontificia Università di Santa Croce di Roma c'è un gruppo di utenti che si occupa di filologia e di edizioni critiche. Essi hanno anche predisposto un corso e un CD-ROM per gli utenti del corso.

15.7 Un esempio di composizione con XeLaTeX

Si vuole qui presentare un breve esempio di composizione in greco classico politonico facendo uso del programma xelatex. Si presenta dapprima il file sorgente e dopo qualche commento il risultato della composizione.

Solo xelatex e lualatex

Dividiamo l'esempio in due parti: (*a*) l'esempio vero e proprio, e (*b*) alcuni commenti in merito alla selezione dei font.

Un'utile introduzione a XeLaTeX, scritta in modo relativamente semplice, è costituita dal testo di Enrico Gregorio, [31]; essa permette di verificare e di approfondire molti degli argomenti esposti qui e di studiarne diversi altri. Esiste anche un manuale iniziale e più o meno ufficiale di XeTeX, [26], che però deve essere ancora considerato sperimentale, almeno non così approfondito come gli altri manuali della serie "The [...] Companion". Un'altra breve guida per XeTeX è stata predisposta da Will Robertson, [56], uno dei collaboratori più importanti alla squadra che si occupa della diffusione di questo programma. In italiano non bisogna assolutamente trascurare il testo [40] scritto apposta per gli umanisti. Per LuaLaTeX i manuali spiegano prevalentemente come servirsi del linguaggio Lua, perché per il resto si comportano come XeLaTeX.

15.7.1 Esempio di composizione in greco

Il file sorgente è stato predisposto con lo *shell editor* TeXShop su una piattaforma Mac nella quale era installato anche il driver per la tastiera greca politonica; le parti in greco sono state scritte direttamente usando la tastiera greca.

Il preambolo è il seguente:

```
%!TEX encoding = UTF-8 Unicode
%!TEX TS-program = xelatex

\documentclass[b5paper,11pt,titlepage]{book}

\setmainfont[Ligatures=TeX,Numbers=OldStyle]{TeX Gyre Bonum}
\setsansfont[Ligatures=TeX,Scale=MatchUppercase]{Futura}

\newfontfamily{\greekfont}{Old Standard}

% Scelta delle lingue

\usepackage{polyglossia}
\setmainlanguage{italian}
\setotherlanguage[variant=ancient]{greek}

\begin{document}
```

Le prime due righe del file specificano a TeXShop l'encoding con cui è stato scritto il file sorgente e il nome del programma con il quale, cliccando il pulsante Typeset, egli deve agire, indipendentemente dalle impostazioni predefinite. Non si insisterà mai abbastanza sulla comodità ed opportunità della presenza delle due righe suddette; TeXShop, TeXworks e TeXstudio interpretano queste righe ed eventualmente modificano temporaneamente le proprie impostazioni;

```
\chapter{Su Archimede}

\section{Testo originale in greco}

\begin{otherlanguage}{\greek}
Οἱόνται τινές, βασιλεῦ Γέλων, τοῦ ψάμμον τὸν
ἀριθμὸν ἄπειρον εἶμεν τῷ πλήθει· λέγω δὲ οὐ μόνον τοῦ περὶ Συρακούσας τε
καὶ τὰν ἄλλαν Σικελίαν ὑπάρχοντος, ἀλλὰ καὶ τοῦ κατὰ πᾶσαν χώραν τάν τε
οἰκημέναν καὶ τὰν ἀοίκητον. ἐντί τινες δέ, οἳ αὐτὸν ἄπειρον μὲν εἶμεν οὐχ
ὑπολαμβάνοντι, μηδένα μέντοι ταλικοῦτον κατωνομασμένον ὑπάρχειν, ὅστις
ὑπερβάλλει τὸ πλῆθος αὐτοῦ. οἱ δὲ οὕτως δοξαζόντες δῆλον ὡς εἰ νοήσαιεν
ἐκ τοῦ ψάμμον ταλικοῦτον ὄγκον συγκείμενον τὰ μὲν ἄλλα, ἁλίκος ὁ τᾶς γᾶς
ὄγκος, ἀναπεπληρωμένων δὲ ἐν αὐτῷ τῶν τε πελαγέων πάντων καὶ τῶν
κοιλωμάτων τᾶς γᾶς εἰς ἴσον ὕψος τοῖς ὑψηλοτάτοις τῶν ὀρέων,
πολλαπλασίως μὴ γνωσόνται μηδένα κα ῥηθήμεν ἀριθμόν ὑπερβάλλοντα τὸ
πλῆθος αὐτοῦ. ἐγὼ δὲ πειρασούμαι τοι δεικνύειν δι' ἀποδειξίων γεομετρικᾶν,
αἷς παρακολουθήσεις, ὅτι τῶν ὑφ' ἁμᾶς κατωνομασμένων ἀριθμῶν καὶ
ἐνδεδομένων ἐν τοῖς ποτὶ Ζεύξιππον γεγραμμένοις ὑπερβάλλοντί τινες οὐ
μόνον τὸν ἀριθμὸν τοῦ ψάμμου τοῦ μέγεθος ἔχοντος ἴσον τᾷ γᾷ
πεπληρωμένᾳ, καθάπερ εἴπαμες, ἀλλὰ καὶ τὸν τοῦ μέγεθος ἴσον ἔχοντος τῷ
κόσμῳ.

\end{otherlanguage}
\end{document}
```

Figura 15.4: Completamento del file sorgente

gli altri editor possono mancare di questa funzionalità o possono averla ma con una sintassi diversa. La persona che ricevesse questo file da altri vedrebbe subito non appena apre il file se eventualmente deve riconfigurare il suo *shell editor* per poter comporre il documento.

Le tre righe dopo la dichiarazione di classe (che è del tutto normale) specificano il modo di dichiarare i font Opentype da usare nel documento. Per il testo in caratteri latini si usa un facsimile del font Bookman attraverso il font della collezione TeX Gyre, già presenti nella distribuzione del sistema TeX. Le opzioni chiedono di usare le legature standard del sistema TeX e chiedono di usare le cifre arabe minuscole (old style).

Il secondo font, Futura, viene usato solo per i caratteri senza grazie; l'opzione serve per scalarlo un pochino affinché le sue lettere minuscole senza ascendenti e discendenti abbiano la stessa grandezza di quelle del font con grazie. Il terzo font, Old Standard, di per sé potrebbe essere usato anche come "mainfont", perché contiene anche i caratteri latini, greci e cirillici, oltre a una quantità di altri segni; ma qui lo si definisce solo come famiglia di caratteri da assegnare alla composizione per la lingua greca; poiché il suo nome `\greekfont` comincia con il nome della lingua, esso viene usato automaticamente come font di default quando si scrive in quella lingua; lo stesso espediente può essere usato per differenziare con la forma o lo stile del font la lingua in cui si sta scrivendo.

Seguono poi i tre comandi per la scelta delle lingue; il pacchetto *polyglossia* sostituisce *babel*; in realtà xelatex e lualatex accettano anche di usare il pacchetto *babel* e il suo modo di specificare le lingue; ma, secondo lo scrivente, è preferibile

usare *polyglossia* perché è già costruito per lavorare con xelatex e lualatex e i loro font OpenType. Inoltre è chiaro qual è la lingua di default, il "mainlanguage" e quali sono le lingue accessorie; qui, in particolare, si specifica il greco con l'opzione che si tratta della variante del greco antico,

Come si vede il preambolo è piuttosto semplice (oltre che esuberante per il piccolo esempio che si sta mostrando).

Il corpo del documento contiene le righe contenute nel medaglione della figura 15.4. Lanciando xelatex o lualatex su questo piccolo file sorgente si ottiene il risultato mostrato nella figura 15.5.

Come si vede chiaramente il font TEX Gyre Bonum propone dei caratteri molto scuri mentre il greco tratto dal font Old Standard è della serie media che si accorderebbe benissimo con i font Latin Modern usati in questo testo. L'accoppiata è quindi infelice, ma la si è mostrata lo stesso per indicare nell'esempio quali generi di font si possono usare con i programmi xelatex e lualatex.

15.7.2 XeLaTeX, i font OpenType e i font Type 1

Quanto si dice qui per XeLaTeX vale anche per LuaLaTeX.

Nel paragrafo precedente abbiamo visto come specificare i font per il testo, come scegliere diversamente i font per una diversa famiglia (sans serif rispetto ai font con grazie) o per una lingua diversa. Quel modo di specificare i font vale anche per i font TrueType se contengono le tabelle con le specificazioni delle varianti come gli OpenType. Secondo gli standard e l'uso corrente, i font Type 1 hanno solo 256 caratteri per ogni polizza, senza varianti e hanno i contorni dei vari segni descritti mediante spline di Bézier di terzo grado. I font TrueType hanno polizze con un numero di segni non limitati a 256, ma possono rispettare le specifiche UNICODE; hanno i contorni dei segni descritti da spline di secondo grado.

Talvolta potrebbe essere necessario specificare l'uso di font Type 1; di questi bisogna specificare la codifica e chiamarli con il loro nome vero, non quello che usa pdflatex; questi nomi sono indicati nel file `pdftex.map` come seconda voce di ogni riga; bisogna stare attenti che possono essere costituiti da diverse parole. Per esempio se uno desiderasse usare i font URW Avant Garde Light Book, la riga corrispondente nel file `pdftex.map` sarebbe:

```
pagk8r URWGothicL-Book "TeXBase1Encoding ReEncodeFont" <8r.enc"<uagk8a.pfb
```

dove `pagk8r` è il nome del file `pagk8r.pfb` che contiene i disegni dei segni di questo particolare font a cui si riferisce il file `pag.fd` di descrizione dei font invocato in definitiva dal pacchetto *avant*. Questo font Type 1 deve essere ricodificato, perché nativamente non ha la codifica *T1*; ecco perché la riga del file `pdftex.map` è così relativamente complessa. Per il nostro caso quello che importa è il nome vero del font da dedurre dal nome compresso `URWGothicL-Book`: il nome completo è `URW Gothic L` e questo bisogna usare nei comandi accettati dal programma xelatex.

Capitolo 1

Su Archimede

1.1 Testo originale in greco

Οἰόνται τινές, βασιλεῦ Γέλων, τοῦ ψάμμον τὸν ἀριθμὸν ἄπειρον εἶμεν τῷ πλήθει· λέγω δὲ οὐ μόνον τοῦ περὶ Συρακούσας τε καὶ τὰν ἄλλαν Σικελίαν ὑπάρχοντος, ἀλλὰ καὶ τοῦ κατὰ πᾶσαν χώραν τάν τε οἰκημέναν καὶ τὰν ἀοἰκητον. ἐντί τινες δέ, οἳ αὐτὸν ἄπειρον μὲν εἶμεν οὐχ ὑπολαμβάνοντι, μηδένα μέντοι ταλικοῦτον κατωνομασμένον ὑπάρχειν, ὅστις ὑπερβάλλει τὸ πλῆθος αὐτοῦ. οἱ δὲ οὕτως δοξαζόντες δῆλον ὡς εἰ νοήσαιεν ἐκ τοῦ ψάμμον ταλικοῦτον ὄγκον συγκείμενον τὰ μὲν ἄλλα, ἁλίκος ὁ τᾶς γᾶς ὄγκος, ἀναπεπληρωμένων δὲ ἐν αὐτῷ τῶν τε πελαγέων πάντων καὶ τῶν κοιλωμάτων τᾶς γᾶς εἰς ἴσον ὕψος τοῖς ὑψηλοτάτοις τῶν ὀρέων, πολλαπλασίως μὴ γνωσόνται μηδένα κα ῥηθήμεν ἀριθμόν ὑπερβάλλοντα τὸ πλῆθος αὐτοῦ. ἐγὼ δὲ πειρασούμαι τοι δεικνύειν δι' ἀποδειξίων γεομετρικᾶν, αἷς παρακολουθήσεις, ὅτι τῶν ὑφ' ἁμῶν κατωνομασμένων ἀριθμὼν καὶ ἐνδεδομένων ἐν τοῖς ποτὶ Ζεύξιππον γεγραμμένοις ὑπερβάλλοντί τινες οὐ μόνον τὸν ἀριθμὸν τοῦ ψάμμου τοῦ μέγεθος ἔχοντος ἴσον τᾷ γᾷ πεπληρωμένᾳ, καθάπερ εἴπαμες, ἀλλὰ καὶ τὸν τοῦ μέγεθος ἴσον ἔχοντος τῷ κόσμῳ.

Figura 15.5: Risultato della composizione con XeLaTeX

In particolare bisogna assicurare che xelatex conosca l'encoding *T1* e che sappia cercare correttamente lo stesso file di descrizione dei font che userebbe pdflatex. Per far questo bisogna specificare attentamente nella giusta successione i comandi seguenti:

```
\usepackage[T1]{fontenc}
\usepackage{fontspec}
\renewcommand{\sfdefault}{pag}
\usepackage{etoolbox}
\cspreto{sffamily }{\fontencoding{T1}}
```

dove

```
\usepackage[T1]{fontenc}
...
\renewcommand{\sfdefault}{pag}
```

sono gli stessi comandi che si darebbero a pdflatex, ma se li si usasse, l'encoding di default sarebbe l'encoding *T1*; se non si specificasse la chiamata a *fontspec*, non si potrebbero usare i font OpenType; per cui è necessario chiamare questo pacchetto, ma a questo punto l'encoding di default diventerebbe quello dei font OpenType, cioè la codifica UNICODE *TU*. Perciò bisogna caricare il pacchetto *etoolbox* che mette a disposizione l'ultimo comando `\cspreto`; questo permette di associare l'encoding *T1* al font specificato solo per la famiglia senza grazie tramite comando `\sfdefault`.

Sembra complicato, e forse la spiegazione lo è davvero, ma in fondo si tratta solo di aggiungere due righe, nell'ordine giusto, a quelle che andrebbero comunque specificate per usare sia i font OpenType in generale sia i font Type 1 particolari che si desiderano.

15.8 Conclusione

Sulla composizione di testi letterari e filologici ci sarebbe da scrivere molto di più di quanto si è esposto in questo capitolo tutto sommato introduttivo nel vastissimo territorio della composizione di questo tipo di documenti.

Si sono indicati pacchetti e programmi di composizione che possono aiutare molto l'umanista nel procedere lungo questo particolare sentiero di composizione tipografica. I pochi esempi mostrati indicano moltissime possibilità di azione. Sta dunque all'umanista, che tanto cura i suoi studi, curare anche l'aspetto compositivo dei suoi lavori. I programmi pdflatex, xelatex e lualatex gli offrono degli strumenti potentissimi di composizione; gli resta quindi il compito di usarli a piena potenza.

Capitolo 16

LaTeX: le presentazioni

16.1 Introduzione

Non è possibile tralasciare, nemmeno in una guida introduttiva, l'argomento delle presentazioni.

Per 'presentazioni' si intendono quelle proiezioni che accompagnano una conferenza, un discorso pubblico, un seminario, una lezione, eccetera.

Se il contenuto delle schermate da proiettare è prevalentemente o completamente testuale, al massimo include delle figure, vanno benissimo i programmi liberi, oppure commerciali, i quali spesso producono proiezioni molto accattivanti, ricche di effetti speciali e piene di parti colorate; spessissimo anche lo sfondo della schermata contiene delle parti disegnate, o delle fotografie, o il logo dell'istituzione di appartenenza dell'oratore; perché non disturbino la lettura queste immagini di sfondo dovrebbero essere molto chiare o comunque alterare il meno possibile il colore dominante dello sfondo.

Tuttavia preparare delle belle presentazioni è un'arte; se poi la presentazione deve contenere dello scritto tecnico (formule matematiche, schemi filologici, o altre cose non facenti parte della prosa 'normale') allora è opportuno usare LaTeX, il che comporta la necessità di saper usare LaTeX al meglio delle sue potenzialità; bisogna conoscere l'arte di comporre con LaTeX oltre che l'arte di predisporre delle schermate efficaci per il trasferimento sintetico dell'informazione dall'oratore all'ascoltatore.

16.2 Le classi per le presentazioni

LaTeX nasce con una semplicissima classe per le presentazioni, `slides`; nonostante la sua semplicità, estensibile facilmente con i vari pacchetti di estensione descritti anche negli altri capitoli, essa contiene alcuni elementi importanti da tenere presenti.

1. Poiché l'ascoltatore recepisce e ricorda meglio quello che legge, che sente, e che vede sullo schermo, è importante che ogni slide, o schermata, contenga un solo concetto e che l'ascoltatore non sia sottoposto a più di un nuovo concetto al minuto.
   In realtà un concetto può essere svolto anche su due o tre slide, ma è preferibile essere sintetici e non è il caso di sommergere l'ascoltatore sotto una valanga di slide, ognuna necessariamente proiettata per poco tempo. Per questo scopo è quindi necessario essere molto sintetici.
2. I due elementi illustrati sopra implicano che ogni slide non deve contenere più di una dozzina di righe di testo (e 12 forse sono già tante) contando anche le righe occupate da espressioni matematiche o da testi in alfabeti speciali o contenenti segni speciali; ognuna di queste righe va forse contata per due!
3. Perciò se la slide viene stampata su fogli di metacrilato (i *lucidi*, chiamati anche impropriamente 'trasparenze') allora i font 'normali' devono essere oltre che facilmente leggibili anche di corpo molto grande. La classe `slides` usa di default per il corpo normale il corpo da 20 pt (per l'esattezza 19,907 pt) della famiglia senza grazie; per migliorare la leggibilità questa famiglia in realtà è una famiglia speciale con l'altezza delle minuscole maggiore del solito: abcdefg; l'esempio ha lo stesso corpo usato nel resto del testo, ma come si vede le sue minuscole sembrano decisamente più grandi; in realtà è più grande la *x-height* ma sono più corti gli ascendenti.
4. Tenendo conto che lo schermo di proiezione ha solitamente il rapporto 4 : 3 fra base ed altezza, anche il foglio di metacrilato o la slide viene stampata facendo riferimento alla base come il lato lungo del foglio e bisogna usare nella dichiarazione di classe l'opzione *landscape*; in questo modo aumenta un poco la giustezza, ma diminuisce l'altezza e il numero di righe disponibile rimane necessariamente basso; non è possibile, perciò mettere troppa informazione in ciascuna slide.
5. La classe `slides` non consente di eseguire animazioni, specialmente se si fa uso dei lucidi di metacrilato; altre classi lo consentono se e solo se non si usano i lucidi, ma si proietta direttamente dal PC o dal laptop mediante un videoproiettore.
   Però con l'uso del pacchetto `color` è possibile colorare lo sfondo e scegliere colori diversi di ogni parte della slide composta con LaTeX, sia testo, sia matematica, sia disegni eseguiti con l'ambiente `picture` o altri ambienti nativi del sistema TeX.

Sia chiaro: oggi è difficilissimo trovare dei proiettori di slide in metacrilato. Oggi la classe `slides` è diventata obsoleta, ma i suoi font non lo sono affatto.

16.3 Altre classi per le presentazioni

Ogni distribuzione del sistema TeX contiene diverse classi per predisporre ottime presentazioni; si possono citare i pacchetti `prosper` e il suo successore `powerdot`,

seminar, *pdfscreen*, *pdfslide*, *ppower4*, *texpower*, per finire con *beamer*.

Ognuno di questi pacchetti produce slide bellissime e le presentazioni di dimostrazione, che accompagnano ogni pacchetto fra la sua documentazione, sono tutte accattivanti.

Ogni pacchetto offre una o più classi per la predisposizione di vari tipi di presentazione; ognuno finisce poi per affezionarsi ad un particolare pacchetto, ma le possibilità di personalizzazione sono tali e tante che non si corre il rischio di eseguire presentazioni con lo stesso stile standard.

16.4 La classe *beamer*

In inglese *beamer* significa videoproiettore. Il pacchetto *beamer* mette a disposizione del compositore sia la classe *beamer* sia una serie di file di estensione modulari con i quali è possibile personalizzare le proprie presentazioni e le modalità per attribuire valori diversi ai diversi elementi che caratterizzano sia il layout delle slide, sia i tipi di comandi o di segni particolari che appaiono nelle slide.

Queste sono concepite per file PDF interattivi, per cui ogni slide contiene, disegnati in colore molto tenue, tanto da non disturbare la lettura, anche i 'pulsanti' per muoversi nella sequenza di slide che costituiscono la presentazione. Inoltre è possibile avere a disposizione una specie di indice generale della presentazione sempre visibile, così che gli ascoltatori possano seguire il cambiamento di colore del titoli delle 'sezioni' che costituiscono la presentazione; questi stessi titoli costituiscono degli *hyperlink* utili all'oratore per navigare rapidamente attraverso la propria presentazione senza avere la necessità di far scorrere la presentazione fino a trovare la slide giusta.

La classe prevede l'uso dei pacchetti *pgf* e *xcolor* così che si possano sfruttare le molteplici funzionalità del pacchetto *pgf* per quel che riguarda la possibilità di eseguire dei bellissimi disegni o per inserire le funzionalità di quel pacchetto per l'inserimento delle immagini esterne.

Invece il pacchetto *xcolor* consente di gestire molto semplicemente i colori per ottenere sia colori misti, sia gradienti di colore lineari o sferici, il tutto con comandi di alto livello molto semplici per l'utente.

Le slide possono avere il loro contenuto esposto incrementalmente; vale a dire che, per esempio, la dimostrazione di un teorema può essere fatta esponendo una dopo l'altra le frasi che ne costituiscono la sequenza logica, senza che il testo già esposto cambi posizione via via che nuovo testo viene esposto. Uno dei vantaggi per l'oratore è che quanto verrà esposto in slide successive appare scritto in colore chiarissimo anche sulle slide precedenti, così che l'ascoltatore quasi non se ne accorge, ma l'oratore ne trae un notevole aiuto perché sa già in anticipo, vedendola, quale sarà la slide successiva.

Ovviamente la classe *beamer* consente di eseguire delle transizioni fra una slide e l'altra agevolando l'ascoltatore nel capire sia i collegamenti fra una slide e la successiva, sia, con transizioni diverse, nel capire quando si passa da una sezione all'altra della presentazione.

La predisposizione delle slide è piuttosto semplice: ognuna corrisponde ad un ambiente *frame*; ogni frame accetta un titolo e il testo della slide; se la slide deve essere presentata in fasi successive, il file di uscita conterrà diverse 'pagine', ma il frame resta invariato; semplicemente i comandi per l'esposizione incrementale delle varie parti della slide fanno parte, nel file sorgente, del testo contenuto nel frame.

Non è possibile in un testo stampato presentare questi effetti dinamici; ma si invita il lettore ad esaminare con attenzione uno dei file 'demo' che accompagna il pacchetto *beamer*; si trovano tutti in `.../doc/latex/beamer/examples/`; suggerisco `beamerexample5.pdf`, ma certi file sono solo per presentare alcuni dettagli, altri sono per presentazioni complete. Un altro esempio è formato dalla coppia di file `beamerexample2.article.pdf` e `beamerexample2.beamer.pdf` derivati dal medesimo file sorgente; a seconda di quali parametri vengono stabiliti nel preambolo, dallo stesso sorgente si possono ricavare sia la presentazione sia un articolo che può costituire direttamente il materiale 'cartaceo' da distribuire all'uditorio.

16.5 La documentazione

Tutti i pacchetti nominati dispongono della loro propria documentazione sul percorso `.../doc/latex/`⟨*pacchetto*⟩`/`. Il file di documentazione di *beamer* si chiama `beameruserguide.pdf` ed è particolarmente interessante, non solo perché contiene una documentazione molto ben fatta, tenendo conto anche della versatilità del pacchetto, e quindi della moltitudine di comandi e ambienti che esso fornisce, ma anche per le preziose informazioni sulla preparazione di presentazioni professionali; questo genere di informazioni non è facilmente reperibile in altra documentazione o in altri libri.

La sostanza è che quasi tutto quello che è opportuno fare da un punto di vista tecnico-compositivo è disponibile con LaTeX e con i comandi messi a disposizione dai pacchetti di servizio *pgf* e *xcolor*, ma quello che deriva dalla professionalità e dal buon gusto non può e non è solitamente descritto da nessuna parte, tranne, appunto, nel manuale di *beamer*.

È sorprendente notare come la maggior parte delle presentazioni composte con noti programmi dedicati, sia commerciali sia freeware, pur essendo quei programmi del tutto validi per lo scopo, peccano perché gli autori non conoscevano e non avevano nessun modo di ricavare dalla documentazione le regole di stile e di buon gusto per la predisposizione di slide efficaci; queste sono efficaci se sono prive di quegli elementi 'decorativi' che, invece di favorire la trasmissione della comunicazione, distraggono lo spettatore/ascoltatore, il quale alla fine della conferenza o della lezione si ricorda di queste animazioni ma non si ricorda di quello che ha detto l'oratore.

È quindi assai raccomandabile leggersi attentamente il manuale di *beamer*; si impareranno moltissime cose che permetteranno di usare al meglio lo strumento

compositivo per ottenere lo scopo di una presentazione, cioè quello di trasmettere un messaggio all'ascoltatore nel modo più efficace possibile.

16.6 Una breve presentazione

Nella figura 16.1 sono rappresentate 8 slide corrispondenti a 5 frame di una brevissima presentazione. Esse sono disposte in verticale; la prima colonna contiene le slide da 1 a 4 e la seconda colonna da 5 a 8.

Il codice per produrre queste slide è il seguente e serve per interpretare il risultato ottenuto nelle slide rappresentate nella figura 16.1.

```
% !TEX encoding = UTF-8 Unicode
%%% file per produrre alcune slide per la GuidaGuit
\documentclass{beamer}
%                                   Preambolo
\usetheme{AnnArbor}
\useoutertheme[right]{sidebar}
\setbeamercolor{alerted text}{fg=red!90!black}
\usepackage[T1]{fontenc}
\usepackage[utf8]{inputenc}
\usepackage{pgf}
\usepackage{curve2e}
\usepackage[color]{guit}
\usepackage[italian]{babel}

\setbeamercovered{transparent}

\usefonttheme{professionalfonts}
\usepackage{lxfonts}

\title{I numeri complessi}
\subtitle{Conferenza internazionale
del~Gruppo~Italiano~degli~utenti~di~\TeX}
\author{Mario Rossi}
\institute{\GuIT}

\date{Pisa, 20--22 ottobre 2015}


\pgfdeclareimage[width=\textwidth]{CifreIndiane}%
                              {CifreIndianeVIIsecolo}
\pgfdeclareimage[width=15.5mm]{guitlogo}{GuITlogo}

\logo{\pgfuseimage{guitlogo}}
% Fine del preambolo
```

Figura 16.1: Otto slide per una presentazione di cinque frame. Vale la pena di osservare che i comandi usati per il diagramma dell'ultima slide sono stati eseguiti con i comandi definiti nel pacchetto *curve2e*.

```
\begin{document}% Inizio della presentazione


\begin{frame}% Primo frame, prima slide
\titlepage
\end{frame}

\begin{frame}% Secondo frame, seconda slide
\frametitle{Indice}
\tableofcontents
\end{frame}

\section{Introduzione storica}

\begin{frame}% Terzo frame, terza slide
\frametitle{Una breve storia dei numeri}
All'inizio del \textsc{vii} secolo gli Indiani inventarono
la notazione posizionale e le <<nove>> cifre dall'1 al 9;
lo zero veniva detto a voce ma non aveva ancora un suo segno.

\begin{center}
\pgfuseimage{CifreIndiane}
\end{center}
\end{frame}

\section{Nascita dei numeri complessi}


\begin{frame}% Quarto frame,  diviso in quattro slide
\frametitle{I numeri complessi nascono nel '500}
\begin{enumerate}
\item<1-> Nel XVI secolo Tartaglia e Cardano introducono
la radice quadrata di $-1$
\item<2-> Il nome di \emph{unità immaginaria} viene creato
da Réné Descartes nel 1637
\item<3-> Gauss nel 1799 contribuisce con i suoi scritti
a diffondere i numeri complessi
\item<4-> Hamilton nel 1833 pubblica la teoria dei numeri
complessi
\end{enumerate}
\end{frame}


\section{I numeri complessi in \texttt{pict2e}}
```

```
\begin{frame}% Quinto frame, ottava slide
\frametitle{I numeri complessi come operatori geometrici}
\begin{center}\unitlength=1mm
   \begin{picture}(60,40)
   \put(0,0){\vector(1,0){60}}\put(60,1){\makebox(0,0)[br]{$x$}}
   \put(0,0){\vector(0,1){45}}\put(1,45){\makebox(0,0)[lt]{$y$}}
   \thicklines
   \put(0,0){\textcolor{red}{\vector(3.4641,2){40}}}
   \thinlines
   \multiput(40,-.5)(0,2){12}{\line(0,1){1}}
   \multiput(-.5,23.094)(2,0){20}{\line(1,0){1}}
   \put(41,1){\makebox(0,0)[bl]{$a$}}
   \put(1,24.1){\makebox(0,0)[bl]{$b$}}
   \VectorArc(0,0)(20,0){30}
   \put(20,4){\makebox(0,0)[bl]{$\varphi$}}
   \put(20,12.6){\rotatebox{30}{\makebox(0,0)[b]{$m$}}}
   \put(0,0){\thicklines\textcolor{blue}{\vector(1,0){10}}}
   \put(5,-1){\textcolor{blue}{\makebox(0,0)[t]{vettore
              unitario}}}
   \end{picture}
\end{center}
Visto come operatore geometrico, il numero complesso
$m\mathsf{e}^{\mathsf{i}\varphi}=a+\mathsf{i}b$ agisce su
un vettore; in questa figura agisce sul
\textcolor{blue}{vettore unitario blu}; lo scala tramite
il fattore \alert{$m$} e lo ruota dell'angolo \alert{$\varphi$}
producendo il \alert{vettore rosso}.
\end{frame}

\end{document}
```

Nel preambolo appaiono diversi comandi, alcuni dei quali sono già familiari, per invocare *babel* con l'opzione *italian* al fine di comporre in italiano; per invocare *inputenc* con l'opzione *utf8* per scrivere usando le lettere accentate; per invocare *fontenc* con l'opzione *T1* per scrivere nel file di uscita con le lettere accentate direttamente, non ottenute per sovrapposizione del segno dell'accento sopra il segno della vocale. Si invoca anche il pacchetto *pgf* per poter usufruire della grafica avanzata, e per la gestione delle figure; il pacchetto *guit* per gestire i loghi del Gruppo degli utilizzatori Italiani di TeX; il pacchetto *curve2e*, che a sua volta carica *pict2e*, per il disegno in uno dei frame della presentazione.

Ma, ai fini di questo esempio, sono interessanti gli altri comandi:

`\usetheme` serve per scegliere il layout generale dei frame, sia come colori, sia come disposizione delle informazioni che corredano la presentazione; in questo caso si è scelto lo stile `AnnArbor` che è uno dei vari stili messi a

disposizione dell'utente di **beamer**; ovviamente questi sono predefiniti, ma l'utente può crearsi dei temi a suo piacimento partendo da zero, oppure può copiare e modificare quelli esistenti mediante gli altri comandi che si vedranno qui di seguito.

`\useoutertheme` serve per scegliere le parti che contornano i frame ai fianchi; in questo caso si è scelto di usare la barra laterale (una striscia con un colore di fondo nella quale appariranno certe informazioni) e si è scelto di mantenerla a destra.

`\setbeamercolor` serve per impostare uno dei colori che beamer usa per certi tipi di testo; in questo caso il colore dell'*allerted text*, il testo su cui richiamare l'attenzione, viene impostato con il 90% di rosso e il restante (10%) nero; insomma si preferisce un rosso un po' più scuro del normale.

`\setbeamercovered` è una dichiarazione che accetta diverse impostazioni con la sintassi:

```
\setbeamercovered{⟨impostazioni⟩}
```

dove ⟨*impostazioni*⟩ può valere: `transparent=`⟨*valore*⟩, `invisible`, `dynamic`, `highly dynamic`, `still covered=`⟨*lista*⟩, `again covered=`⟨*lista*⟩; gli argomenti sono essenzialmente delle impostazioni booleane il cui significato è piuttosto chiaro; ma quelli che accettano una ⟨*lista*⟩ sono piuttosto delle opzioni del tipo ⟨*opzione*⟩=⟨*valore*⟩; che cosa significhi ⟨*lista*⟩ è bene approfondirlo nella documentazione di **beamer**. Il ⟨*valore*⟩ di trasparenza è prefissato a `0.85`, ma l'utente può esplicitamente specificarne un altro. Con l'impostazione `transparent` si chiede che il testo da esporre in slide successive sia 'trasparente', molto chiaro in modo da intravederlo anche quando il testo non è esposto.

`\title` `\author` e `\date` sono vecchie conoscenze, ma **beamer** consente di specificare anche un sottotitolo per la presentazione mediante il comando `\subtitle`. Il modo particolare con cui è scritto il sottotitolo serve per garantire che la frase sia divisa in due sottofrasi in modo che ognuna contenga le sue congiunzioni e preposizioni semplici o articolate.

`\institute` serve per inserire il nome o il logo dell'istituzione a cui appartiene l'oratore.

`\pgfdeclareimage` è un comando che consente di assegnare non solo certe caratteristiche all'immagine da usare, in questo caso la larghezza, ma anche un nome con cui richiamarle insieme alle loro proprietà. Attenzione, non si tratta di una cosa da poco; l'immagine `GuITlogo` viene riprodotta in ogni slide, quindi se ogni volta si ricaricasse il file del disegno, si correrebbe il rischio di finire con file dalle dimensioni enormemente grandi; usando invece `\pgfdeclareimage` il particolare file che contiene l'immagine viene caricato una sola volta grazie al fatto che il programma pdflatex è in grado di riutilizzare quell'immagine un numero illimitato di volte senza copiarla ma facendovi riferimento mediante i suoi meccanismi interni di gestione degli hyperlink. Questa funzionalità è disponibile anche se si usa lualatex o

xelatex; salvo casi particolari dove sia necessario usare font OpenType, si suggerisce di usare ogni volta che si può il compilatore pdflatex.

`\logo` dice quali comandi devono venire usati per produrre l'immagine del logo nel punto in cui questo deve apparire.

`\pgfuseimage` infine è il comando che, riferendosi al nome di un'immagine precedentemente dichiarato mediante `\pgfdeclareimage`, effettivamente crea gli hyperlink interni che rendono visibile l'immagine nel punto voluto.

`\usefonttheme` con l'argomento `professionalfonts` informa *beamer* che si intende usare una collezione di font completa, vale a dire che non contiene solo i font testuali, ma anche i corrispondenti font matematici; chi scrive usa sempre i font della collezione LXfonts che, appunto, sono quelli usati in questi esempi e sono completi dei corrispondenti font matematici. Questi font sono una ricostruzione estesa dei font usati dalla classe *slides*, particolarmente leggibili e privati di alcune lettere che potevano essere confuse con altre[1]. Con il comando `\usepackage{lxslides}` si ridefiniscono tutte le impostazioni per i font testuali e matematici, permettendo l'uso anche dei caratteri greci sia nel testo sia in matematica.

Con l'inizio del documento appaiono le dichiarazioni dei vari frame mediante i successivi ambienti *frame*. Il primo frame contiene il titolo della presentazione; il comando `\titlepage` fa sì che il frame sia composto in modo un po' differente dagli altri, ma che contenga buona parte delle informazioni relative al titolo, al presentatore, e alla sua istituzione di appartenenza, all'evento nell'ambito del quale si svolge la presentazione, eccetera; la prima slide e primo frame della figura 16.1 mostra tutti questi dettagli. Nell'angolo in basso a destra compare anche il numero progressivo del frame riferito al numero complessivo dei frame che compongono la presentazione; nell'esempio compare la frazione 1/5 per dire che si tratta del primo di cinque frame.

Il secondo frame contiene l'indice della presentazione; è utile che questo indice venga presentato all'inizio, perché è un semplice mezzo per consentire all'oratore di delineare all'uditorio il piano della sua presentazione. Questo stesso elenco è collegato internamente agli hyperlink che consentono di navigare all'interno della presentazione; le stesse voci di questo indice compaiono anche nella barra laterale destra, e si illuminano di un colore più chiaro ogni volta che si cominciano ad esporre le varie slide che compongono una data sezione; gli ascoltatori seguendo questa barra possono rendersi conto istante per istante del punto a cui è arrivato l'oratore, senza bisogno di controllare l'orologio con mosse ampie e vistose per farsi notare dall'oratore affinché ci dia un taglio... Spero che nessuno dei lettori di queste note si trovi mai a fare o a subire queste azioni!

Tra il secondo frame e il terzo è inserito un comando di sezionamento; il suo contenuto consente di creare l'indice, ma non compare mai al di fuori dell'indice; è bene che una presentazione sia strutturata in sezioni, eventualmente anche in sottosezioni, ma non si frammenti troppo l'esposizione; durante un

[1] I caratteri che potevano ingenerare confusione sono stati ridisegnati in modo da evitare questo inconveniente; quindi non si tratta degli *slide font* originali, ma di loro derivati.

congresso è possibile che ogni oratore abbia a sua disposizione 20 minuti di cui cinque sono destinati a rspondere alle domande poste dagli ascoltatori; quindi la presentazione non dovrebbe contenere più di 15–20 slide; due o tre sezioni sono più che sufficienti per strutturare la presentazione. Per una lezione, magari di due ore, è evidente che si può strutturare maggiormente, ma anche in questa circostanza non è il caso di fare un unico file con un indice interminabile; gli allievi si demoralizzerebbero, specialmente tenendo conto che durante una lezione assistita da una presentazione, quanto viene detto è molto di più di ciò che si potrebbe esporre con le sole parole o con l'aiuto di una lavagna su cui scrivere o disegnare. Una simile lunga lezione è meglio che sia strutturata in due parti (due distinti file) da presentare con un congruo intervallo fra le due.

Si vede che nel terzo frame è richiamata per nome una delle due immagini precedentemente dichiarate e si fa uso di `\pgfuseimage` che svolge il ruolo che normalmente avrebbe il comando `\includegraphics` del pacchetto *graphicx*.

Il quarto frame corrisponde a 4 slide; si vede che esso contiene una enumerazione, ma dopo ogni comando `\item` compare una strana espressione: due parentesi 'acute' racchiudono un intervallo nel quale sono indicate le slide da proiettare; il primo termine dell'elencazione deve essere proiettato dalla slide 1 in poi; il secondo elemento dalla slide 2 in poi, eccetera; la sintassi di questo intervallo di numerazione delle slide segue questa regola: se le parentesi acute contengono un solo numero, senza il trattino, la slide viene proiettata solo quando tocca a lei (in base al numero d'ordine); se invece l'espressione contiene un intervallo del tipo `2-3` il termine dell'elencazione appare solo per le slide da 2 a 3 comprese, ma non compare mentre viene proiettata la slide numero 1, né quando viene proiettata la numero 4. Se manca un numero prima del trattino, si intende il numero 1; se manca un numero dopo il trattino, si intende fino all'ultima slide di questo frame. Il fatto, però, è che le slide dello stesso frame hanno tutte lo stesso contenuto; ma quelle che devono mostrare solo una parte del contenuto, ce l'hanno in colore trasparente molto chiaro, quasi invisibile per chi non sa che cosa c'è scritto, ma non per l'oratore che invece intravede quanto apparirà nella slide successiva.

Infine nel quinto frame c'è un disegno eseguito con i comandi dell'ambiente *picture*, come esteso dal pacchetto *curve2e*, che, lo si ripete, carica il pacchetto *pict2e*.

16.7 Creare un nuovo stile

Gli stili di presentazione nel gergo di *beamer* si chiamano *temi*, più precisamente con la parola inglese *theme*; la cartella di *beamer* contiene diverse sottocartelle per le configurazioni di diversi temi, separati in modo modulare, in modo che cambiare un tema, non vuol dire cambiare gli altri. Se un autore vuole costruirsi uno stile di presentazione proprio, lo può fare in modi diversi; il più semplice, forse, è quello di vedere sulla documentazione di *beamer* i vari temi già predisposti, scegliere quello che più si avvicina allo stile desiderato e poi modificarlo caricando

dopo quel tema, gli altri moduli tematici; uno per i colori, uno per le barre, uno per i campi da inserire nelle riga di testa e nella righe in calce, eccetera.

L'altra via è quella di leggere con attenzione la documentazione relativa ai temi e scrivere autonomamente le macro che occorrono; non è semplice, ma non è nemmeno troppo difficile.

Il modo più semplice è quello di ricorrere al sito: `http://titilog.free.fr/indexEng.htm`; in questo sito c'è solo una finestra con una slide di modello; si clicca sulle varie parti che si vogliono personalizzare scegliendo dal menù sul lato destro sia i colori dello sfondo, si quelli dei font da usare, eccetera. Alla fine si clicca il grosso bottone ovale contenente la scritta *Finish* e il gioco è fatto; sotto alla slide modello c'è il bottone di download, per scaricare direttamente il file `.sty` contenente il tema appena creato. Il nome di questo file deve cominciare con la locuzione `beamertheme` e deve proseguire con una parola o locuzione mnemonica, per esempio il nome del congresso durante il quale la presentazione viene esposta ai congressisti, per esempio `ecoc2010`; il file del tema avrà quindi il nome complessivo: `beamerthemeecoc2010.sty`. Cliccando sul bottone *Download* si esegue lo scaricamento del file sul proprio disco.

Per usare questo nuovo tema basta inserire nel preambolo

```
\usetheme{ecoc2010}
```

Ovviamente, dopo aver creato il nuovo tema nessuno vieta di eseguire delle altre modifiche agendo direttamente sul file, ma il grosso è già fatto in modo grafico interattivo, quindi il tema dovrebbe essere già a posto e usabile immediatamente.

16.8 Osservazioni

Questo capitolo serve per richiamare l'attenzione del lettore sull'opportunità di documentarsi a fondo nelle rispettive guide in merito alle potenzialità dei vari pacchetti citati. Alcuni, come *beamer*, sono nati per produrre il risultato finale direttamente in formato PDF, altri, come *prosper* e il suo discendente *powerdot*, ricorrono al pacchetto *PSTricks* e quindi devono subire due trasformazioni: prima devono essere composti con latex per ottenere il file DVI, poi questo deve essere trasformato in formato PDF attraverso il doppio passaggio dvips e ps2pdf[2]. Il gioco può valere la candela, come si dice, ma dipende dal tipo di disegni che è necessario inserire nelle slide.

È inevitabile che ognuno si affezioni ad una particolare classe, anche perché bisogna usare molti nuovi comandi dai nomi talvolta non così facili da ricordare; perciò l'abitudine diventa un fattore importante.

Però parlando di arte di comporre con LaTeX, non si può negare che LaTeX è uno strumento formidabile, ma da solo non è sufficiente per predisporre presentazioni efficaci e accattivanti. La necessità di documentarsi nell'arte della comunicazione orale e visiva diventa essenziale; padroneggiando bene quest'arte diventa poi abbastanza facile usare lo strumento LaTeX con i suoi pacchetti di estensione in modo da ottenere il meglio.

[2]Il programma dvipdfm non è in grado di interpretare il codice PostScript complesso generato dai comandi di *PSTricks*.

Capitolo 17

LaTeX: la microgiustificazione

17.1 Introduzione

Quando si vogliono fare bene le cose, meglio ancora di come le farebbe LaTeX da solo, che rasenta già la perfezione in fatto di composizione tipografica, bisogna ricorrere alle potenzialità di LaTeX, mediante i compilatori pdflatex e lualatex, che non sono documentate né nel libro/manuale di Leslie Lamport, né in molti altri autorevoli libri, ma sono riportate solo nella documentazione che accompagna il pacchetto *microtype*. Il programma xelatex per ora può sfruttare la microgiustificazione in modo parziale, come d'altra parte succede anche con latex; questo è dovuto al passaggio intermedio per il formato DVI (semplice o esteso).

Si tratta di questo: la composizione tipografica di qualità richiede tra l'altro che il testo composto appaia privo di 'ruscelli' e sia di un 'colore' uniforme. I *ruscelli* tipografici sono quelle sequenze di spazi interparola che si susseguono in diverse righe successive in corrispondenza quasi perfetta gli uni con gli altri, così che essi formano una striscia bianca che serpeggia attraverso il testo come un ruscelletto bianco.

Non solo ma gli spazi interparola dovrebbero essere il più omogenei possibile e non troppo grandi, in modo che il testo, visto da una distanza alla quale non si riesca a leggere quanto esso contiene, appaia di un colore (grigio) uniforme.

Gli artisti tipografi sanno che lo spazio interparola ideale dovrebbe essere quello di una lettera 'i' nel font nel quale è composto il testo. Knuth, invece, nel suo TeXbook afferma che lo spazio interparola ideale sia largo quanto una 'e' nel font corrente. Probabilmente non esiste un valore di spazio da prescrivere come ideale, perché la spaziatura piccola della 'i' può andare bene per un testo letterario, quello di una 'e' probabilmente è un buon compromesso in generale, ma potrebbero esserci dei testi specifici, specialmente di carattere scientifico, dove la spaziatura interparola diventa importante per separare bene il testo,

dalla matematica, dalle parole tecniche, eccetera. Quello che è certo è che la composizione con noti word processor e la composizione in colonne strette praticata nelle tipografie di molti quotidiani e molti settimanali rinuncia a questa omogeneità degli spazi interparola producendo risultati 'terribili' nonostante i mezzi tecnici a disposizione del compositore.

A questi problemi pone rimedio il pacchetto *microtype* che offre soluzioni modeste a latex e xelatex, ma che sviluppa tutta la sua forza con pdflatex e lualatex.

17.2 La composizione dei capoversi

Il TeXbook spiega dettagliatamente come l'interprete pdftex componga il testo sorgente in capoversi giustificati. I dettagli sono piuttosto complicati, ma sotto sotto c'è tutta l'abilità del matematico Knuth che ha prodotto un algoritmo di ottimizzazione capace di dividere il capoverso in righe giustificate minimizzando la somma totale dei *demerits* associati alle righe in cui sarebbe possibile dividere un capoverso; questi *demerits* sono legati quadraticamente alla *bruttezza* della divisione scelta fra le divisioni possibili oltre che ad altri elementi costituiti generalmente da *penalità*.

La *bruttezza* è un indice definito a partire dall'ammontare dell'allargamento o del restringimento dello spazio interparola nelle righe giustificate dell'intero capoverso. Le penalità sono presenti esplicitamente, per averle specificate nel testo sorgente, o anche indirettamente, perché sono state introdotte dai comandi usati; alcune penalità derivano dall'aver inserito divisioni di parole in fin di riga. Le *penalità* da usare nei diversi contesti sono predefinite nel file di formato e in quelli di classe, ma l'utente può inserirne di altre attraverso comandi come `\break`, `\nobreak`, `\linebreak`, e simili, oppure inserendo alcune cesure esplicite mediante il comando `\-`.

Altri *demerits* sono ulteriormente introdotti dal programma di composizione se la penultima riga di un capoverso riceve la cesura. Il pacchetto *babel*, per la lingua italiana, impone il valore di 50 000 000 al parametro `\finalhyphendemerits` che controlla, appunto, la cesura della penultima riga di un capoverso. In italiano, infatti sarebbe quanto mai indesiderabile che venga divisa la penultima riga di un capoverso, consentendo che nell'ultima riga ci possa essere solo un frammento di parola, al limite due sole lettere, corrispondenti alla minima lunghezza di una sillaba terminale secondo l'algoritmo di sillabazione di LaTeX per l'italiano. Quell'enorme valore di *demerits* non è 'infinito', ma consente praticamente sempre che l'ultimo righino di un capoverso cominci con una parola intera.

Tutte le tabelle dei font 18.3–18.12 contengono anche i parametri metrici caratteristici dell'intera polizza, in particolare l'ammontare dello spazio interparola e il suo allungamento e il suo accorciamento; per esempio nella tabella 18.5, relativa al font a 10 pt usato per comporre questo testo, si legge che lo spazio interparola è pari a 3,3333 pt, che l'allungamento vale 1,6666 pt e che l'accorciamento vale 1,1111 pt. Questo significa che nel giustificare le righe TeX aggiusta gli

spazi interparola aggiungendo o togliendo allo spazio 'normale', pari a un terzo del corpo, quanto consentito dall'allungamento o dall'accorciamento previsti, aggiungendo alla bruttezza una quantità dipendente dal cubo dell'ammontare di questi allungamenti o accorciamenti relativi rispetto ai valori naturali[1].

Quando si inserisce il comando di legatura fra due parole o due informazioni tra le quali si vieta la divisione della riga, quando, insomma, si usa il carattere attivo `~`, per ottenere questo effetto, il comando inserisce una penalità pari a 10 000 (che per TEX è sinonimo di 'infinito') per cui una eventuale divisione in quel punto provocherebbe una bruttezza infinita, al punto che qualunque altra divisione produrrebbe una bruttezza inferiore.

Le cesure in fin di riga vengono minimizzate ricorrendo a due distinte composizioni; il programma esegue una prima divisione delle righe giustificate senza dividere in sillabe e calcola la (minima) bruttezza del capoverso così ottenuto. Se questo valore è inferiore a quello conservato in `\pretolerance`, allora questa divisione viene accettata e le righe giustificate del capoverso vengono accodate alla pagina parzialmente composta. Se questo valore di `\pretolerance` viene superato, allora il programma esegue una seconda divisione delle righe consentendo la cesura in fin di riga; esso però attribuisce una penalità ad ogni cesura introdotta, una penalità ad ogni sequenza di due righe consecutive contemporaneamente contenenti la cesura, e una ulteriore dose di *demerits* se la penultima riga del capoverso contiene la cesura. Tutte queste penalità contribuiscono ad incrementare i *demerits* della divisione in righe; fra tutte le possibili combinazioni di divisioni, con le corrispettive cesure e i corrispettivi allungamenti o accorciamenti degli spazi interparola, il programma sceglie quella che produce la minima bruttezza complessiva e la confronta con un altro valore conservato in `\tolerance`. Se questa minima bruttezza non supera `\tolerance`, la divisione in righe viene accettata, altrimenti TEX ci prova una terza volta ma il risultato finale viene accodato alla pagina in composizione accompagnato dai messaggi relativi alla bruttezza e alla presenza di righe 'overfull' o 'underfull'; le prime sporgenti fuori della giustezza, e le seconde con un allargamento eccessivo degli spazi interparola.

Questo fatto accade molto raramente; nonostante tutto può accadere. Se l'eccesso di lunghezza non supera 1 pt–1,5 pt, spesso si può chiudere un occhio, ma se si può, è meglio provvedere. Se lo spazio interparola viene allargato troppo ma la bruttezza della riga non supera qualche migliaio (1000–2000) si può chiudere un occhio, ma di nuovo se si può rimediare, è meglio provvedere. Generalmente questo implica la modifica del testo. In linea di massima, senza ricorrere alla microgiustificazione, spetta al compositore, d'accordo con l'autore, il compito di modificare il testo del capoverso affinché la sua bruttezza sia ridotta; certamente il programma è molto potente, ma non arriva a modificare il testo da solo!

[1] La funzione di bruttezza legata a questi allungamenti o accorciamenti relativi è sempre cubica, ma è diversa per l'allungamento, che può teoricamente assumere qualunque valore, rispetto all'accorciamento che non può essere superiore al valore 'normale', altrimenti una parola si sovrapporrebbe alla precedente.

17.3 Metodi per migliorare la giustificazione

Dai paragrafi precedenti si evince chiaramente che latex 'normale' può ricorrere solamente all'aggiustamento dello spazio interparola e alla cesura in fin di riga per comporre i capoversi.

I tipografi che lavoravano a mano con i caratteri mobili avevano anche altre tecniche per giustificare meglio i capoversi.

spazieggiatura In inglese *tracking*. Spazieggiare i caratteri significa comporli con uno spazio interlettera leggermente maggiore di quello di default. Per i più autorevoli *book designer* britannici spazieggiare è paragonato all'abigeato (furto di bestiame). Per noi il paragone è quasi incomprensibile, ma la sua forza risiede nel fatto che il furto di bestiame era un reato molto grave nella Gran Bretagna del XIX secolo, tanto che veniva punito con la pena di morte.

Oggi si conviene che la spazieggiatura sia accettabile e talvolta consigliabile nella composizione in lettere maiuscole o maiuscolette di titoli o di simili brevi tratti di testo; assolutamente improponibile in un testo in lettere minuscole. Il lettore attento provi ad esaminare una qualunque rivista pubblicata ai nostri giorni e cerchi di determinare quali righe siano spazieggiate; constaterà che la spazieggiatura non solo è visibile ad occhio nudo, ma anche che contribuisce non poco a rendere mal composto il capoverso che contenga tali righe spazieggiate. Egli constaterà anche che la raccomandazione di non spazieggiare il testo scritto in lettere minuscole è abbondantemente disattesa in quel genere di pubblicazioni; probabilmente una composizione giustificata solo a sinistra sarebbe molto più raccomandabile.

protrusione della punteggiatura La protrusione della punteggiatura è quella tecnica che consente di comporre con il margine destro leggermente seghettato, invece che perfettamente allineato, ma in modo che le parti sporgenti siano costituite solo da segni di punteggiatura, segnatamente il punto e la virgola, oltre che da una frazione del trattino di cesura. L'occhio, data la scarsa forza visiva di questi segni, vede il margine destro giustificato anche se in effetti esso non lo è esattamente.

protrusione dei segni grafici Con lo stesso criterio si possono far uscire piccole parti dei segni alfabetici, numerici, di punteggiatura, eccetera, fuori da entrambi i margini, purché questa protrusione ammonti ad una frazione piccolissima del segno che protrude. Questo implica che fra le caratteristiche metriche del font deve essere indicata per ciascun segno una coppia di valori numerici che rappresentano la parte millesimale della larghezza del singolo segno che può protrudere nel margine di sinistra o, rispettivamente, nel margine di destra. Spesso nei font con grazie ci si può accontentare di far sporgere nei margini piccole porzioni delle grazie orizzontali.

larghezza del font Si può ancora ricorrere a varianti del font corrente formate da font dello stesso corpo, forma e nerezza, ma leggerissimamente più 'estesi' o 'compressi'. Una variazione di larghezza del $\pm 2\%$ dovrebbe essere

considerata come la variazione massima otticamente accettabile affinché l'occhio del lettore non si accorga della variazione.

legature e segni diacritici Nel comporre si potrebbero usare font dotati di segni legati in numero maggiore dei soliti ff, fi, ffi, fl, ffl e ß presenti nei font delle tabelle 18.3 e 18.5. Gli antichi usavano molte altre legature, presenti oggi solo nei font esperti, e ricorrevano a segni diacritici, come per esempio l'uso della tilde sulla lettera precedente per marcare una 'm' o una 'n' omessa: non è raro trovare scritture come *librũ* al posto di *librum*. Oggigiorno questo genere di artifici non viene più usato in italiano. In tedesco, invece, la umlaut sopra certe vocali è il residuo di questa pratica e rappresenta una 'e'; infatti quando in tedesco bisogna scrivere con font che mancano della umlaut, si mette esplicitamente la 'e' a destra della vocale che non ha potuto ricevere la umlaut: *Universität* o *Universitaet*; *Müller* o *Mueller*.

17.4 La microgiustificazione

I prototipografi ricorrevano spesso alle tecniche illustrate sopra; Gutenberg stesso avrebbe usato varianti più larghe o più strette del font 'normale' per comporre una delle sue famose Bibbie. Gutenberg stesso non avrebbe usato allargamenti o restringimenti maggiori del 2%, tanto che sarebbero state necessarie misure molto accurate da parte degli studiosi di tipografia antica per rendersi conto del 'trucco' usato al fine di comporre così bene.

I metodi descritti nel paragrafo precedente possono essere usati anche con i programmi di composizione del sistema TEX.

Segnatamente la protrusione della punteggiatura può essere usata con latex e xelatex, mentre questa protrusione e le altre tecniche possono essere tutte usate con i programmi pdftex e lualatex.

La scelta, quindi, di usare la piena potenzialità dei meccanismi di microgiustificazione o soltanto la protrusione dei segni di interpunzione dipende da altri fattori; come si è detto, ma conviene ripeterlo, latex e xelatex sono limitati alla protrusione dei segni, mentre pdflatex e lualatex non hanno questa limitazione ma possono servirsi sia della protrusione sia dell'espansione dei font.

Detto ciò, la microgiustificazione si ottiene richiamando il pacchetto *microtype*. Questo pacchetto riconosce da solo con quale programma viene composto il particolare documento e si comporta conseguentemente per usare solo le funzionalità di cui il programma è capace.

Ovviamente il pacchetto *microtype* è configurabile mediante opzioni passate al comando con il quale lo si richiama; alcune funzionalità possono essere attivate o disattivate dal compositore a seconda di che cosa egli sta componendo.

Non solo, ma adeguatamente istruito con comandi da inserire nel preambolo o lungo il file sorgente, il pacchetto è in grado di cambiare impostazioni a seconda del contesto nel quale sta operando. Per esempio nelle note a piè di pagina le opzioni da usare potrebbero essere diverse da quelle che sono in azione durante la composizione del testo; per esempio, in francese, dove la spaziatura dei segni

di interpunzione è particolare, potrebbero essere in vigore le opzioni per lo spazio extra prima dei segni di interpunzione, oppure all'interno delle parti delimitate dalle virgolette caporali, mentre in italiano queste opzioni devono essere disabilitate; in un documento scritto in italiano ed in francese le spaziature extra prima dei segni di interpunzione apparirebbero solo nelle parti scritte in francese.

Le possibilità di gestire tutte le funzioni di microgiustificazione sono numerose; quindi si rinvia il lettore alla documentazione in `.../doc/latex/microtype/microtype.pdf`.

Questo testo è stato composto con l'uso del pacchetto *microtype*; con l'aiuto di un righello e una versione stampata di almeno alcune pagine del testo è possibile verificare la protrusione dei segni di interpunzione e la protrusione nei due margini di piccole parti di diversi caratteri. L'espansione o la contrazione dei caratteri è difficile da misurare; tuttavia è possibile che la stessa parola compaia in righe diverse con differenti livelli di compressione o di espansione; misurando accuratamente la larghezza di tali parole si può verificare la presenza di questo particolare tipo di microgiustificazione.

Si noterà anche che le parole con cesura in fin di riga sono proprio poche; generalmente si tratta di parole lunghe almeno otto caratteri, spesso decisamente più lunghe; gli spazi interparola non sono proprio uguali, ma sono molto più omogenei di quelli che si avrebbero senza l'uso del pacchetto.

Disponendo dei file sorgente di questo testo si può provare a vedere che cosa si ottiene senza usare il pacchetto *microtype*; si noteranno spazi interparola più disomogenei, molte più righe con cesure in fin di riga, e molti più messaggi di `overfull` oppure di `underfull hbox`.

Si osservino comunque nella pagina 391 le tre composizioni dello stesso testo a confronto; la prima è stata ottenuta componendo un capoverso sufficientemente lungo su due colonne e con la microgiustificazione attivata; la seconda composizione contiene le stesse parole, ma la microgiustificazione è disattivata; nella terza composizione, invece, la microgiustificazione è attiva solo per quel che riguarda la protrusione, mentre è disattivata l'opzione per ricorrere alle varianti del font con larghezze diverse. Si tenga presente che all'interno dell'ambiente *multicols*, con cui sono eseguite le tre composizioni, le penalità per la presenza di cesure in fin di riga sono drasticamente ridotte, proprio per tener conto della composizione in colonne strette.

Nel secondo testo, non microgiustificato, si osserva una riga in più, spazi interparola più grandi e si vedono chiaramente alcuni ruscelli, mentre non ne appaiono nel testo microgiustificato; il testo non microgiustificato contiene anche più cesure in fin di riga e diverse sequenze di righe consecutive terminanti con la cesura. Nel terzo testo si ha una situazione intermedia, ma è tutto quello che si può ottenere quando il documento è composto per l'uscita in formato DVI.

Oggi si conviene che la spazieggiatura sia accettabile e talvolta consigliabile nella composizione in lettere maiuscole o maiuscolette di titoli o di simili brevi tratti di testo; assolutamente improponibile in un testo in lettere minuscole. Il lettore attento provi ad esaminare una qualunque rivista pubblicata ai nostri giorni e cerchi di determinare quali righe siano spazieggiate; constaterà che la spazieggiatura non solo è visibile ad occhio nudo, ma anche che contribuisce non poco a rendere mal composto il capoverso che contenga tali righe spazieggiate. Egli constaterà anche che la raccomandazione di non spazieggiare il testo scritto in lettere minuscole è abbondantemente disattesa in quel genere di pubblicazioni; probabilmente una composizione giustificata solo a sinistra sarebbe molto più raccomandabile.

Oggi si conviene che la spazieggiatura sia accettabile e talvolta consigliabile nella composizione in lettere maiuscole o maiuscolette di titoli o di simili brevi tratti di testo; assolutamente improponibile in un testo in lettere minuscole. Il lettore attento provi ad esaminare una qualunque rivista pubblicata ai nostri giorni e cerchi di determinare quali righe siano spazieggiate; constaterà che la spazieggiatura non solo è visibile ad occhio nudo, ma anche che contribuisce non poco a rendere mal composto il capoverso che contenga tali righe spazieggiate. Egli constaterà anche che la raccomandazione di non spazieggiare il testo scritto in lettere minuscole è abbondantemente disattesa in quel genere di pubblicazioni; probabilmente una composizione giustificata solo a sinistra sarebbe molto più raccomandabile.

Oggi si conviene che la spazieggiatura sia accettabile e talvolta consigliabile nella composizione in lettere maiuscole o maiuscolette di titoli o di simili brevi tratti di testo; assolutamente improponibile in un testo in lettere minuscole. Il lettore attento provi ad esaminare una qualunque rivista pubblicata ai nostri giorni e cerchi di determinare quali righe siano spazieggiate; constaterà che la spazieggiatura non solo è visibile ad occhio nudo, ma anche che contribuisce non poco a rendere mal composto il capoverso che contenga tali righe spazieggiate. Egli constaterà anche che la raccomandazione di non spazieggiare il testo scritto in lettere minuscole è abbondantemente disattesa in quel genere di pubblicazioni; probabilmente una composizione giustificata solo a sinistra sarebbe molto più raccomandabile.

17.5 Come funziona la microtipografia

Il pacchetto *microtype* è formidabile nel realizzare le funzionalità descritte nei precedenti paragrafi, ma non funziona da solo. Perché il pacchetto possa svolgere il suo lavoro, bisogna che possa esercitare un controllo sui font che fanno parte del documento da comporre.

Questo implica che tutti i font del documento devono essere corredati di file adeguati per informare *microtype* delle caratteristiche di ogni segno e/o di ogni gruppo di segni.

Il pacchetto *microtype* dispone di un file di configurazione, `microtype.cfg`, nel quale sono contenute le impostazioni predefinite per alcuni insiemi di font, di alcune codifiche, di alcune famiglie, di alcune serie, di alcune forme e di alcuni corpi. Se l'utente vuole usare font che non appartengono agli insiemi predefiniti, può caricare il pacchetto con l'opzione *config* ma per i suoi font speciali deve predisporre file di configurazione che si devono chiamare `mt-`⟨*famiglia*⟩`.cfg`; per esempio, volendo usare i font MinionPro, allora bisogna predisporre un file di configurazione con il nome `mt-MinionPro.cfg`.

I file di configurazione devono specificare una quantità di informazioni molto dettagliate relative alle varie codifiche, famiglie, serie, forme, corpi dei font a cui si applicano e ai singoli caratteri o insiemi di caratteri presenti nel font stesso; queste informazioni riguardano sia l'ammontare di quanto ogni carattere o insieme di caratteri affini può protrudere nei margini di sinistra o di destra; sia l'ammontare di quanto ogni carattere può essere allargato o contratto in relazione ai millesimi della sua larghezza naturale. Si tratta di informazioni molto dettagliate sia nel significato di ogni misura millesimale specificata, sia in relazione al fatto che i caratteri, anche se raggruppati in insiemi di caratteri affini, sono molto numerosi.

L'utente deve essere conscio che senza queste descrizioni le funzionalità disponibili con il pacchetto *microtype* non sono usabili per determinati font, a maggior ragione per font personali, o acquistati o, comunque, scaricati dalla rete se sono gratuiti e di uso non riservato. Tutto è possibile, ma la costruzione di file di configurazione per *microtype* relativi a font particolari sono di costruzione molto delicata, quindi molto difficile. Il lettore è inviato a leggere con molta attenzione la documentazione del pacchetto *microtype* con il solito comando da terminale `texdoc microtype`.

Tuttavia queste parole non scoraggino chi vuole usare *microtype* con font di cui non dispone dei file di configurazione. Certo le informazioni metriche relative a ciascun glifo o glifi affini (per esempio: a, à, á, â,...) potrebbero essere migliori se fossero accuratamente calibrate per ogni collezione specifica di font. Infatti le impostazioni disponibili per i font preimpostati sono dei ragionevoli compromessi, perché non hanno nessuna caratteristica 'eccessiva'; per esempio i font con grazie non hanno grazie molto sporgenti, né grazie appena visibili; sono una giusta via di mezzo, per una microgiustificazione valida anche con altri font. È ovvio che se ogni raccolta di font avesse i file specifici per *microtype* il lavoro di questo pacchetto sarebbe ancora migliore, ma in mancanza bisogna fare di necessità virtù, e accontentarsi del lavoro già buono che *microtype* riesce a fare anche in questi casi.

Si esamini con cura la pagina 394, dove l'esempio della pagina 391 è ripetuto usando i font del Johannes Kepler Project (file `kpfonts`), invece dei Latin Modern.

Il primo blocco è composto con la piena funzionalità di *microtype*, cioè con la protrusione e la compressione dei font; il secondo blocco è composto senza né protrusione né compressione dei font; il terzo blocco è composto con la sola protrusione dei font. Come si vede il primo e il terzo blocco risultano composti

nello stesso modo, mentre il secondo è visibilmente differente, con spazi maggiori fra le parole e con un numero maggiore di cesure in fin di riga e con una riga in più nella composizione dell'intero capoverso. I font JKP non dispongono del loro file di configurazione, quindi le informazioni metriche sono quelle di default. Nonostante questa mancanza, il risultato con la piena funzionalità del pacchetto *microtype*, o anche con la sola protrusione, è decisamente migliore che senza tale funzionalità.

Oggi si conviene che la spazieggiatura sia accettabile e talvolta consigliabile nella composizione in lettere maiuscole o maiuscolette di titoli o di simili brevi tratti di testo; assolutamente improponibile in un testo in lettere minuscole. Il lettore attento provi ad esaminare una qualunque rivista pubblicata ai nostri giorni e cerchi di determinare quali righe siano spazieggiate; constaterà che la spazieggiatura non solo è visibile ad occhio nudo, ma anche che contribuisce non poco a rendere mal composto il capoverso che contenga tali righe spazieggiate. Egli constaterà anche che la raccomandazione di non spazieggiare il testo scritto in lettere minuscole è abbondantemente disattesa in quel genere di pubblicazioni; probabilmente una composizione giustificata solo a sinistra sarebbe molto più raccomandabile.

Oggi si conviene che la spazieggiatura sia accettabile e talvolta consigliabile nella composizione in lettere maiuscole o maiuscolette di titoli o di simili brevi tratti di testo; assolutamente improponibile in un testo in lettere minuscole. Il lettore attento provi ad esaminare una qualunque rivista pubblicata ai nostri giorni e cerchi di determinare quali righe siano spazieggiate; constaterà che la spazieggiatura non solo è visibile ad occhio nudo, ma anche che contribuisce non poco a rendere mal composto il capoverso che contenga tali righe spazieggiate. Egli constaterà anche che la raccomandazione di non spazieggiare il testo scritto in lettere minuscole è abbondantemente disattesa in quel genere di pubblicazioni; probabilmente una composizione giustificata solo a sinistra sarebbe molto più raccomandabile.

Oggi si conviene che la spazieggiatura sia accettabile e talvolta consigliabile nella composizione in lettere maiuscole o maiuscolette di titoli o di simili brevi tratti di testo; assolutamente improponibile in un testo in lettere minuscole. Il lettore attento provi ad esaminare una qualunque rivista pubblicata ai nostri giorni e cerchi di determinare quali righe siano spazieggiate; constaterà che la spazieggiatura non solo è visibile ad occhio nudo, ma anche che contribuisce non poco a rendere mal composto il capoverso che contenga tali righe spazieggiate. Egli constaterà anche che la raccomandazione di non spazieggiare il testo scritto in lettere minuscole è abbondantemente disattesa in quel genere di pubblicazioni; probabilmente una composizione giustificata solo a sinistra sarebbe molto più raccomandabile.

Capitolo 18

LaTeX: i caratteri e i font

18.1 Introduzione

I caratteri da stampa, o meglio, i segni che i caratteri imprimono sulla carta ci sono familiari perché abbiamo una grande abitudine a leggere testi scritti a stampa; tuttavia siamo assolutamente ignoranti per quanto riguarda la loro storia, l'evoluzione, gli usi, i periodi storici, e quant'altro riguardi questi piccoli oggetti e i corrispondenti piccoli segni.

Tuttavia si sono già dette alcune cose in merito ai caratteri, che qui brevemente si richiamano.

18.2 Terminologia relativa ai caratteri

font è praticamente sinonimo di 'carattere' con l'unica avvertenza che in italiano si usa la parola 'carattere', specialmente al plurale, per designare una intera polizza di segni accomunati da una omogeneità stilistica, ma si usa anche, specialmente al singolare, per designare il singolo segno. Font è equivalente al primo significato, quello di 'polizza', mentre il singolo segno in inglese viene designato *glyph*; anche in italiano si potrebbe dire glifo, ma è insolito vedere usata questa parola[1].

polizza indicava l'intera cassa di caratteri accomunati dalle stesse caratteristiche grafiche e tipografiche, incluso il numero esatto dei caratteri metallici presenti in ogni scomparto della cassa. Oggi che i caratteri metallici si usano assai poco e solo per lavori molto particolari, non ha più significato

[1] Curiosamente quando si usano parole inglesi, per lo più di genere neutro, in italiano, dove il neutro non esiste, i diversi parlanti usano il genere grammaticale che più li ispira, senza un criterio particolare. La parola *font* è una di quelle che non sembra ancora essersi stabilizzata su un determinato genere grammaticale. Qui si usa il maschile (che in italiano spesso è usato anche come neutro, come genere grammaticale indifferente); spesso si sente dire e si vede scritto *le font*, però *la font* è una espressione rarissima. In ogni caso, visto che font ormai è entrato nell'italiano, ricordiamoci che in questa lingua non prende la 's' del plurale inglese.

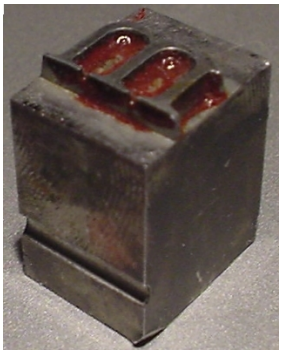


Figura 18.1: Il carattere metallico della ‘m’; il disegno, ovviamente, è come allo specchio e la sinistra e la destra sono scambiate

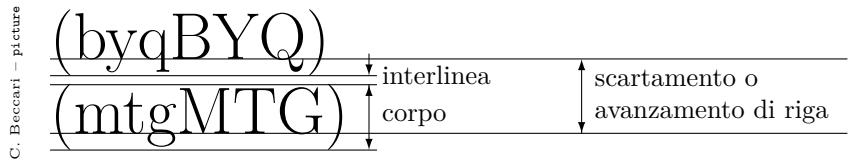


Figura 18.2: Relazioni fra il corpo, l’interlinea e l’avanzamento di riga nel caso di caratteri elettronici

usare la parola ‘polizza’ per designarne anche la consistenza, in quanto questa è illimitata con i caratteri elettronici, ma la parola viene usata solo per indicare una collezione di segni; quasi sempre viene sostituita dall’equivalente parola inglese *font*, al punto che questa parola si può considerare definitivamente entrata nell’uso della lingua italiana.

corpo è la dimensione reale per i caratteri metallici, e quella virtuale per i caratteri elettronici, dell’altezza del rettangolo costituito dalla faccia in rilievo del parallelepipedo che costituiva il carattere mobile metallico. Nella figura 18.1 è riportato uno di questi blocchetti e vi si può riconoscere il disegno in rilievo e rovesciato di una ‘m’.

interlinea era una striscia di metallo di un dato spessore da inserire fra due righe di testo composte con i caratteri mobili; per estensione, spazio verticale da aggiungere all’altezza di una riga composta con caratteri elettronici. Oggi questo nome viene impropriamente usato al posto della locuzione corretta “avanzamento di riga” o “scartamento”, per cui diventa necessario usare il termine “interlineatura” per indicare l’operazione di distanziamento di due righe di testo, quell’operazione che un tempo veniva eseguita con l’inserimento dell’interlinea metallica.

sterlineatura è il processo opposto a quello dell’interlineatura; interlineando si allontanano le righe fra di loro, sterlineando le si accosta. È evidente che questa operazione non poteva essere fatta con i caratteri metallici, perché

[1.20]LA RIVOLUZIONE FRANCESE
LA PRESA DELLA BASTIGLIA

[0.94]LA RIVOLUZIONE FRANCESE
LA PRESA DELLA BASTIGLIA

Figura 18.3: Composizione normale e sterlineata

non li si poteva compenetrare gli uni negli altri; al massimo potevano venire limati, o costruiti apposta, ma era una pratica costosissima e la si evitava. Con i caratteri metallici si diceva che le linee erano sterlineate quando non veniva inserita fra di loro nessuna interlinea; componendo in corpo 10/10, per esempio, si soleva dire che il testo era sterlineato.

Con i caratteri elettronici, al contrario, le righe possono venire accostate fino anche a sovrapporsi; può essere una tecnica interessante per ottenere effetti speciali in stampati pubblicitari. Nei testi di cui ci stiamo occupando, invece, la sterlineatura viene usata specialmente quando si compone con le sole lettere maiuscole, le quali sono praticamente prive di discendenti; a causa di ciò la composizione in maiuscolo con lo scartamento abituale appare troppo aperta. Nella figura 18.3 c'è un esempio di sterlineatura; nella parte superiore il testo è composto in corpo 25/30, mentre nella parte inferiore è sterlineato, composto in corpo 25/23,5, vi è, cioè, una interlineatura negativa.

Va da sé che la scelta dell'interlinea o dell'ammontare della sterlineatura va fatta con molto buon senso e con molto buon gusto da parte del grafico editoriale; dipende molto dal tipo di carattere, dalla sua serie e dalla sua forma. In generale la composizione in sole lettere maiuscole oppure in maiuscoletto e maiuscole consente delle operazioni che non sono lecite o praticabili con gli alfabeti maiuscoli/minuscoli a causa della forma delle lettere e in particolare per l'assenza o della presenza dei discendenti.

avanzamento di riga è la distanza fra le righe di base di due righe di testo composte con caratteri dello stesso corpo. Si può pensare a questa grandezza come alla somma del corpo e dell'interlinea; numericamente si tratta dello stesso valore, ma questa somma parte dalla base del rettangolo della faccia

porta carattere, mentre l'avanzamento di riga parte dalla linea di base e raggiunge la linea di base della riga precedente; si veda la figura 18.2.
In Italia lo *scartamento*, o*avanzamento di riga*, viene spesso chiamato *riga* e i *tipometri* spesso sono graduati in righe e servono per misurare sia sullo stampato sia sulla gabbia della composizione in caratteri metallici proprio lo scartamento; spesso hanno diverse scale con righe di 12 punti, di 10 punti, eccetera. In sostanza l'espressione *riga di 12 punti* in Italia viene usata con il significato di *pica* nei paesi anglosassoni o di *cicero* nell'Europa continentale. Qui si eviterà accuratamente di usare la parola 'riga' in questo significato, perché è un termine troppo ambiguo.

scartamento è sinonimo di avanzamento di riga; richiama alla mente il significato ferroviario come distanza fra i binari; in tipografia è la distanza fra due linee di base di due righe di testo consecutive composte con lo stesso font; il parallelismo di queste due linee di base e quello dei due binari ferroviari è l'elemento che giustifica questo termine.

x-height è l'altezza delle lettere minuscole senza ascendenti né discendenti, cioè 'a c e m n o r s u v w x z'; questa altezza definisce anche l'unità di misura 'ex' che permette di prendere le misure in termini di altezza dei font che sono in uso. Il termine italiano per x-height è *occhio*.

quadro o quadrato era un blocco metallico quadrato usato come spaziatore nel comporre con i caratteri mobili; i due lati del quadrato erano lunghe quanto il corpo del font in uso, se non altro perché l'altezza del quadrato doveva essere uguale al corpo. Con i font elettronici si conviene che la dimensione di ciascun lato del quadrato sia pari alla larghezza della 'M' per cui si usa parlare della distanza di una M, 'em' in inglese. Se si leggono i dati forniti in calce alle tabelle 18.3–18.12, si vede che 1 em spesso non coincide affatto con il corpo anzi ne è sensibilmente diverso. Nonostante ciò questa spaziatura di un quadrato continua a fare riferimento all'unità em. In italiano il blocchetto si chiamava quadrato, e un blocchetto rettangolare largo il doppio di un quadrato si chiamava 'quadratone'; i comandi per realizzare queste spaziature si sono già incontrati negli esempi matematici; una spaziatura di un quadrato si ottiene con il comando `\quad` e di un quadratone con il comando `\qquad`.

legatura è l'operazione per la quale, per ottenere il giusto effetto estetico, certe lettere distinte devono essere accostate in modo così forte che vengono a formare un unico segno; si pensi a 'fi', 'ff', 'ffi', 'fl', 'ffl' presenti in ogni font latino; ma queste non sono le sole legature; scrivere `--` produce – e scrivere `---` produce —; si tratta sempre di legature; anche la lettera tedesca ß è il risultato della legatura di una 's lunga' e di una 's corta', in definitiva di due lettere 's' di forma diversa. In Italia non si usa più questo segno da alcuni secoli, ma i prototipografi italiani nel Quattrocento e nel Cinquecento la usavano abbondantemente; l'uso è cessato con la sparizione della 's lunga' nel diciottesimo secolo; questo segno assomigliava ad una lettera 'f', solo che la barretta non attraversava il fusto; se si esamina il segno (corsivo) della *ß*, si riconosce che la parte di sinistra è proprio una

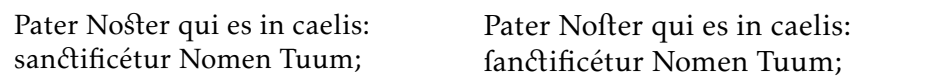


Figura 18.4: Legature antiche; il testo a sinistra mostra solo le legature dei nessi 'ct' e 'st'; quello di destra mostra la compresenza della s lunga e dalla s corta e delle legature antiche.

's lunga'. Questa forma della lettera 's' veniva usata sempre come lettera iniziale o come lettera mediana, mentre la 's corta', quella che usiamo oggi, era in posizione terminale o come secondo elemento della doppia 's', da cui la legatura. Spesso nel passato erano legati anche i nessi 'ct' e 'st'. Nella figura 18.4 sono riportati due brevi testi che mostrano l'uso delle legature antiche e della s lunga[2].

grazie sono quei piccoli tratti alla fine delle aste spesse o sottili che permettono, in particolare, di allineare esattamente i singoli caratteri sulla linea di base, riempiono leggermente lo spazio fra un carattere e l'altro e danno l'idea di una linea più compatta, così che l'occhio la segue meglio e ritrova più facilmente l'inizio della riga seguente quando l'occhio deve andare a capo. I caratteri con grazie sono più adatti alla lettura continua, mentre i caratteri senza grazie affaticano il lettore se egli dovesse leggere con continuità molte pagine di seguito. Invece i caratteri senza grazie, a pari corpo, apparentemente sono più facilmente leggibili a distanza, tanto che i cartelli stradali sono composti esclusivamente con caratteri senza grazie. Non è sempre facile classificare i caratteri senza grazie; per esempio il font Optima, disegnato da Hermann Zapf, pur non avendo grazie, almeno nel senso tradizionale della parola, non può essere definito *sans serif* in virtù della sua particolare conformazione ad aste svasate che lo avvicina ai font dotati di grazie.

sans serif è l'espressione inglese per designare i font senza grazie; in italiano si chiamano preferibilmente caratteri 'lineari', o 'bastone', o 'bastoncino'; si usa anche l'aggettivo 'sgraziato', ma questa parola si presta a interpretazioni diverse.

maiuscoletto è una raccolta di segni dove l'alfabeto è composto dalle lettere maiuscole e da una serie di 'minuscole' costituite da segni con lo stesso disegno delle maiuscole ma di altezza e larghezza minori; non si tratta di maiuscole rimpicciolite, ma di lettere disegnate con le stesse caratteristiche grafiche; per esempio hanno lo stesso spessore delle aste corrispondenti; una 'A' rimpicciolita alle dimensioni di una 'a' appare così: 'a'; affiancate appaiono così: aa; si può percepire che la 'a' rimpicciolita è leggermente più chiara della a maiuscoletta.

[2]Si è scelto il testo latino anche per poter mostrare la legatura del nesso 'ct'. Il testo usa gli accenti secondo la consuetudine del latino ecclesiastico. Inoltre, rispetto alla versione della Vulgata, l'aggettivo *cotidianus* sostituisce l'originale *transubstantialis* secondo la lezione moderna.

proporzionale è il tipo di carattere che destina uno spazio diverso a ciascuna lettera; come ognuno può constatare la larghezza della 'm' minuscola è decisamente più grande della 'i' minuscola; non confrontiamo nemmeno la differenza fra maiuscole e minuscole perché è troppo evidente. La maggior parte, per non dire la totalità, dei caratteri usati in tipografia sono caratteri proporzionali. Tuttavia esiste ed è utilissimo, specialmente nei testi dove si parla di informatica, poter disporre di un font a spaziatura fissa come quello usato in questo testo per descrivere i comandi usati per il mark-up di LaTeX; si osservino i due alfabeti maiuscolo e minuscolo sovrapposti:

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
```

per rendersi conto di che cosa sia un carattere non proporzionale confrontandolo con i due corrispondenti alfabeti di un font proporzionale:

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz

famiglia rappresenta una collezione di font di diversi disegni ma con una caratteristica comune; le famiglie più importanti per LaTeX sono quella dei caratteri a spaziatura fissa, chiamati *typewriter type*, che costituiscono la famiglia identificata dall'istruzione `\ttfamily` o dal comando `\texttt`; la famiglia dei font proporzionali senza grazie, *sans serif*, identificata dall'istruzione `\sffamily` o dal comando `\textsf`; la famiglia dei font proporzionali con grazie, *roman*, identificata dall'istruzione `\rmfamily` o dal comando `\textrm`.

serie è la caratteristica di ogni famiglia di font di apparire più o meno scura; la tipografia tradizionale prevede diversi livelli di nero che vanno dal chiarissimo al nerissimo; per LaTeX si distinguono tre livelli di nero; il medio, il nero e il nero esteso; di norma il nero 'semplice' è assai raro nelle famiglie di caratteri usate di default da LaTeX, perciò le istruzioni per scegliere queste caratteristiche di default sono solamente `\mdseries` e `\bfseries` oppure i comandi `\textmd` e `\textbf`; non tutte le famiglie dispongono di serie più scure di quella che è considerata media.

forma rappresenta il tipo di disegno di una serie di caratteri della stessa famiglia e della stessa serie; per LaTeX le forme disponibili normalmente sono la forma diritta, *upright*, la forma corsiva, *italics*, e quella maiuscoletta, *small caps*; le istruzioni per scegliere queste forme sono `\upshape`, `\itshape` e `\scshape`, mentre i comandi sono `\textup`, `\textit` e `\textsc`. Sono disponibili anche la forma tondo inclinato, *slanted*, e il corsivo diritto, *upright italics*; queste in Italia sono meno usate, tuttavia LaTeX mette a disposizione l'istruzione `\slshape` e il comando `\textsl` per il tondo inclinato. Invece per il corsivo diritto non sono disponibili né l'istruzione né il comando, sebbene non sia difficile servirsi di questa forma con comandi di livello più basso.

Si confronti il tondo inclinato con il corsivo: *abcd* e *abcd*; si nota che si tratta di due disegni (diversamente) inclinati, ma sono completamente diversi; questo è il motivo per il quale il tondo inclinato è poco usato. Il corsivo diritto appare così: abcd.

codifica è la speciale modalità informatica con la quale un particolare segno di un dato font viene identificato nel file che contiene tutti i segni di quel particolare font. LaTeX conosce di default diverse codifiche, le più usuali per chi scrive in italiano sono la codifica `OT1` e la codifica `T1`; la prima identifica i file che contengono 128 caratteri 'latini', la seconda identifica i file che contengono 256 caratteri 'latini'; i font 'latini' contengono oltre alle cifre, ai segni di interpunzione e ad un certo numero di segni speciali, gli alfabeti minuscolo e maiuscolo delle 26 lettere 'latine':

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz

I numerosi segni accentati che compaiono nelle varie lingue che usano i caratteri latini sono ottenuti sovrapponendo l'accento alla lettera con la codifica `OT1`, mentre sono ottenuti con i disegni delle lettere già accentate con la codifica `T1`.

Per capire meglio il concetto di codifica si esaminino le tabelle 18.3–18.12 che contengono i caratteri latini con codifica OT1, i caratteri latini con codifica T1, i caratteri non letterali del Text Companion Font con codifica TS1, i caratteri cirillici con codifica OT2, i caratteri greci con codifica LGR. In ogni casella di quelle tabelle c'è il segno che corrisponde alla posizione indicata dal numero decimale che compare nella stessa casella; i numeri preceduti da un solo apice o dal doppio apice nelle righe e nelle colonne esterne rappresentano parti degli indirizzi in notazione ottale o in notazione esadecimale; per esempio nella tabella 18.5 la lettera 'Í' ha un indirizzo decimale 205; questa casella si trova all'incrocio della riga con indirizzo ottale `'300` e della colonna con indirizzo ottale `'15` da cui si ricava l'indirizzo ottale complessivo eseguendo la somma dei due valori ottali: `'300 + '15 = '315`. Nella stessa maniera si ricava l'indirizzo complessivo esadecimale. Nei file interni di LaTeX sono quasi sempre usati gli indirizzi ottali o quelli esadecimali; quindi è conveniente disporre di tabelle che contengano tutti e tre i tipi di indirizzi.

Se si confrontano le tabelle 18.3 e 18.5, che si riferiscono entrambe ai caratteri latini, oltre alle ovvie differenze relative alle lettere accentate, ci sono alcuni segni uguali che si trovano ad indirizzi diversi; per esempio il segno 'ffi' si trova all'indirizzo 14 con la codifica OT1, ma all'indirizzo 30 con la codifica T1.

È assai raro dover cambiare codifica per comporre un testo; per questo LaTeX non dispone di una istruzione o di un comando di uso corrente per eseguire questo cambio di codifica, ma questo tipo di azione viene svolto dietro le quinte dalle istruzioni contenute nei file di servizio, che provvedono direttamente a cambiare l'alfabeto e la codifica a seconda della lingua che

Normale con pdflatex; raramente con lualatex o xelatex

si usa. L'unico momento in cui il compositore decide di usare una delle due codifiche per i caratteri latini è all'inizio del file nel preambolo, quando ordina di usare il *font encoding* T1, la codifica T1, mediante la specifica

```
\usepackage[T1]{fontenc}
```

Se si desiderasse cambiare codifica in una posizione qualunque del testo bisognerebbe usare comandi di basso livello come `\fontencoding` e `\selectfont`, scrivendo per esempio

```
\fontencoding{T1}\selectfont
```

possibilmente all'interno di un gruppo per limitarne gli effetti al solo testo contenuto nel gruppo.
In ogni caso se in un file sorgente da comporre con xelatex o lualatex si dovesse per forza usare un font Type 1 con le codifiche descritte in questo paragrafo, bisognerebbe seguire una precisa sequenza di chiamata dei pacchetti e dei font: si veda il medaglione della sintassi nella pagina 402.

```
% Sequenza di comandi per usare un font Type 1
% in un file sorgente da compilare con XeLaTeX
% o LuaLaTeX
...
\usepackage[utf8]{inputenc}
\usepackage[⟨T1⟩]{fontenc}
\usepackage{⟨pacchetto per usare un dato font Type 1⟩}
\usepackage{fontspec}[⟨opzioni per i font OpenType⟩]
...
\begin{document}
...
{\usefont{T1}{⟨famiglia⟩}{⟨serie⟩}{⟨forma⟩}
⟨testo⟩
\par}
...
```

Ovviamente ci si possono definire comandi appositi per usare la giusta sequenza di font di basso livello. Si noti che la ⟨*famiglia*⟩ dei font Type 1 invocati con il pacchetto specifico è definita nei suoi file con estensione `.fd`. Ma da questo esempio si capisce anche come ci si deve muovere per usare in font Type 1 insieme ai font OpenType usati da xelatex e lualatex. Ci si domanda se il gioco valga la candela, però, anche se questa procedura si può evitare quasi sempre per comporre testi 'normali' potrebbe essere necessario ricorrervi per produrre testi dove, per esempio, si mostrano font codificati in modi diversi.

tipo Importantissimo per i font da usare con la tipografia assistita da calcolatore, la tipografia elettronica, è il tipo dei caratteri. Questi possono essere di tipo a *matrice di punti* o *bitmapped* o *raster*, oppure *vettoriali*. La differenza è essenziale.

I caratteri a matrice di punti sono una versione digitalizzata o discretizzata del disegno del singolo segno; ogni segno è formato da una schiera di puntini 'neri' sufficientemente accostati in modo da rendere l'idea abbastanza precisa del disegno del segno specifico. I puntini vanno benissimo se la loro densità corrisponde esattamente alla densità dei puntini della macchina da stampa o dello schermo con cui il carattere deve venire stampato o visualizzato. Non sono suscettibili di ingrandimenti o rimpicciolimenti significativi, altrimenti il segno si sgrana e si notano i vari 'quadratini' che compongono il disegno. Vanno quindi usati solo per la stampa.

(*a*) (*b*)

Figura 18.5: Il simbolo di *maschio* fortemente ingrandito: (*a*) font a matrice di punti; (*b*) font vettoriale.

I font vettoriali, invece, sono descritti dalle equazioni del loro contorno; la discretizzazione per la stampa o per lo schermo viene eseguita al volo dal programma che trasforma l'informazione codificata nel disegno del segno in un insieme di puntini neri visibili sulla carta o sullo schermo. L'ingrandimento e il rimpicciolimento del disegno vengono ottenuti cambiando semplicemente scala alle coordinate dei punti che individuano i contorni. In particolare i nodi su cui sono 'appoggiati' i vari archi che formano i contorni si possono assimilare alle punte dei vettori che uniscono l'origine ai nodi stessi; di qui la denominazione di disegni *vettoriali*.

La differenza sembra quindi nel fatto che i segni a matrici di punti sono memorizzati come matrice di puntini neri, mentre i segni vettoriali sono trasformati in matrice di puntini neri solo al momento dell'uso.

Non è così, almeno non lo è quando il supporto di 'stampa' è lo schermo; in realtà non è così nemmeno quando il supporto è la carta, ma la differenza è molto maggiore quando si considera lo schermo.

Chi legge un testo direttamente da uno schermo tende ad adattare l'ingrandimento della pagina sia alle dimensioni dello schermo fisico sia al tipo di dettaglio che desidera osservare. Si consideri allora la figura 18.5.

Si vede chiaramente, specialmente lungo il contorno, la presenza dei quadratini neri nel disegno (*a*) che è a matrice di punti, ma non si vede nessuna

granulosità nel contorno del disegno (*b*) che è vettoriale.
In pratica, con il sistema TeX i font originariamente prodotti con il programma METAFONT, quando la grafica degli schermi non era granché, sono a matrici di punti; il programma METAFONT è però sofisticatissimo ed è in grado di generare delle matrici di punti anche con una densità adeguata ad una macchina di fotocomposizione professionale, con densità di punti di più di 4000 punti al pollice; inoltre METAFONT può tenere conto anche di densità diverse in orizzontale rispetto alla direzione verticale.
Con l'avvento dei primi font vettoriali da parte della Adobe, cioè dei font PostScript Type 1, la tecnologia si è sviluppata moltissimo[3]; con una iniziativa di collaborazione fra Apple e Microsoft vennero sviluppati poi i font vettoriali TrueType; successivamente sono stati sviluppati anche i font OpenType, sempre vettoriali, ma molto più completi dei Type 1 e dei TrueType. Con i programmi del sistema TeX i font OpenType possono essere usati direttamente solo da xelatex e lualatex, oltre che da ConTeXt MK IV. Nel seguito si darà molta importanza ai font vettoriali e si descriveranno molte azioni per poter rendere disponibili all'uso anche i font Type 1 e TrueType commerciali o anche quelli non ancora completamente integrati nella distribuzione del sistema TeX.

18.3 I comandi per la scelta dei font

Per scegliere i font il compositore dispone di una serie di istruzioni e di comandi che sono già stati quasi tutti esposti nel paragrafo precedente. Vale però la pena di riepilogare anche per distinguere sistematicamente i vari comandi e le corrispondenti dichiarazioni a seconda del contesto in cui si usano.

18.3.1 La scelta del corpo e dello scartamento

LaTeX dispone di una serie di comandi che si riferiscono al corpo dei caratteri in modo simbolico, invece che numerico; essi sono raccolti nella tabella 18.1 e indicano tutti quanti (in inglese) la relazione che il corpo prescelto ha rispetto al corpo normale.

Il corpo che funziona da corpo normale è quello di default, cioè di 10 pt, oppure quello che si specifica fra le opzioni del comando di apertura `\documentclass`; fra queste opzioni, in generale, ci sono solo *11pt* e *12pt*. Alcune classi non standard offrono anche font normali di corpo più piccolo o più grande.

Bisogna rilevare una differenza sostanziale fra i font; ne esistono di due tipi diversi; (*a*) quelli con i corpi ottici e (*b*) quelli in un corpo unico. La differenza

[3]Incidentalmente sono classificati font di tipo Type 3 i font ottenuti per accostamento di tante parti distinte con possibili sovrapposizioni di alcune parti; i font bitmapped, quando vengono inclusi in un documento PDF, vengono classificati come Type 3. Si ricordi che per qualunque operazione di stampa più o meno commerciale un documento che faccia uso di font Type 3 viene generalmente respinto.

Istruzione	Esempio
`\tiny`	ABCDEFG abcdefg
`\scriptsize`	ABCDEFG abcdefg
`\footnotesize`	ABCDEFG abcdefg
`\small`	ABCDEFG abcdefg
`\normalsize`	ABCDEFG abcdefg
`\large`	ABCDEFG abcdefg
`\Large`	ABCDEFG abcdefg
`\LARGE`	ABCDEFG abcdefg
`\huge`	ABCDEFG abcdefg
`\Huge`	ABCDEFG abcdefg

Tabella 18.1: Istruzioni per la scelta del corpo dei caratteri

sostanziale è la seguente e riguarda prevalentemente i font vettoriali, ma non è una loro esclusività. I font vettoriali, proprio perché la loro natura permette di scalarli con continuità, vengono solitamente distribuiti in un corpo unico e vengono ingranditi o rimpiccioliti con i comandi della tabella 18.1 partendo dalla loro unica realizzazione fisica; questa è generalmente di corpo 10 pt, talvolta leggermente più grande, ma queste informazioni sono conservate nel file metrico che li accompagna e che il programma di composizione interpreta correttamente per inserirli nel documento finale nel corpo richiesto.

Invece il primo tipo di font, dotato di corpi ottici, dispone di una varietà di disegni leggermente diversi da corpo a corpo in modo che quelli più piccoli del font "normale" non diventino troppo fini e leggeri, con il pericolo di perdere qualche dettaglio molto fine, e quelli più grandi del normale non appaiano troppo neri.

Nella tabella 18.2 si vede direttamente l'effetto che i comandi di corpo producono su font dotati di corpi ottici e su quelli che vengono distribuiti in un solo corpo. nella prima riga di ogni coppia appare il font in corpo 10 scalato al corpo specificato nella seconda colonna; mentre nella seconda riga di ogni coppia appare il font nella sua forma come prescritta dall'apposito comando di corpo. Si vede chiaramente che i font in corpo unico hanno lo stesso disegno ingrandito o rimpicciolito sia che si scalino sia che si usino i comandi di corpo. Invece i font dotati di corpi ottici hanno un disegno diverso per i vari corpi, per cui scalando

Latin Modern	
abcdefghijklmnopqrstuvwxyz abcdefghijklmnopqrstuvwxyz	Font scalato a 8pt `\footnotesize`
abcdefghijklmnopqrstuvwxyz abcdefghijklmnopqrstuvwxyz	Font scalato a 9pt `\small`
abcdefghijklmnopqrstuvwxyz abcdefghijklmnopqrstuvwxyz	Font scalato a 12pt `\large`
abcdefghijklmnopqrstuvwxyz abcdefghijklmnopqrstuvwxyz	Font scalato a 14.4pt `\Large`
abcdefghijklmnopqrstuvwxyz abcdefghijklmnopqrstuvwxyz	Font scalato a 17.28pt `\LARGE`
Times eXtended	
abcdefghijklmnopqrstuvwxyz abcdefghijklmnopqrstuvwxyz	Font scalato a 8pt `\footnotesize`
abcdefghijklmnopqrstuvwxyz	Font scalato a 9pt
abcdefghijklmnopqrstuvwxyz	`\small`
abcdefghijklmnopqrstuvwxyz abcdefghijklmnopqrstuvwxyz	Font scalato a 12pt `\large`
abcdefghijklmnopqrstuvwxyz abcdefghijklmnopqrstuvwxyz	Font scalato a 14.4pt `\Large`
abcdefghijklmnopqrstuvwxyz abcdefghijklmnopqrstuvwxyz	Font scalato a 17.28pt `\LARGE`)

Tabella 18.2: Confronto fra l'ingrandimento di un font dotato di corpi ottici con un font dotato di un solo corpo

verso corpi minori l'alfabeto si accorcia e le linee diventano troppo sottili, mentre scalando verso corpi maggiori l'alfabeto si allunga e le linee diventano troppo spesse così che il font sembra nero invece che di serie media.

In questa tabella infatti si vede che cosa succede ingrandendo o rimpicciolendo un font di corpo 10 pt oppure regolandone il corpo con i comandi della tabella 18.1;

nella seconda metà della tabella si vede che cosa succede con con un font come il Times eXtended dotato di un solo corpo; l'ingrandimento e l'uso dei comandi di corpo producono esattamente lo stesso risultato. Invece nella prima metà della tabella l'uso dei comandi di corpo sui font Latin Modern dotati di corpi ottici produce il risultato voluto, mentre l'ingrandimento o il rimpicciolimento produce un risultato diverso: più piccolo nel caso del rimpicciolimento al punto di rendere illeggibili i corpi minori, e più grande e più nero nel caso dell'ingrandimento per i corpi maggiori.

Questo fatto deve essere tenuto in conto quando si scelgono font diversi da quelli predefiniti. Si tenga presente che quasi tutti i font vettoriali commerciali distribuiti sia insieme al sistema TEX (gratuitamente usabili senza restrizioni) sia quelli acquistati preso i produttori (con licenze variamente onerose) sono generalmente prodotti in un solo corpo. Fanno eccezione alcuni di quelli nati con il sistema TEX e alcuni font molto professionali, e molto costosi, che spesso vengono distribuiti solamente in quattro corpi ottici.

Ancora un dettaglio: i font Latin Modern sono corredati da file di descrizione che consentono di ingrandire e rimpicciolire i font a corpi qualunque; l'operazione viene fatta sul corpo ottico di valore più vicino a quello richiesto, affinché la variazione di corpo rispetto a quello di partenza sia la minima possibile. In questo modo i "difetti" prodotti dal cambiamento di scala sono ridotti al minimo e sono del tutto impercettibili. Sul modello di questo file di descrizione sono stati costruiti anche quelli dei font greci predefiniti (dotati di corpi ottici) da usarsi quando i font latini corrispondenti sono i Latin Modern.

I corpi più piccoli di quello normale si differenziano di un punto l'uno dall'altro; i corpi più grandi sono invece in proporzione geometrica approssimativamente di ragione 1,2. Per il corpo normale di 10 pt i corpi specificati con i comandi della tabella 18.1 si susseguono così: 5 pt, 7 pt, 8 pt, 9 pt, 10 pt, 12 pt, 14,4 pt, 17,28 pt, 20,74 pt, 24,88 pt. Fuori della sequenza in progressione geometrica c'è anche il corpo di 10,95 pt, usato come corpo normale con l'opzione di *11pt*; si noti che questo corpo approssima la media geometrica fra i valori adiacenti 10 pt e 12 pt: $10{,}95 \approx \sqrt{10 \times 12}$.

Gli scartamenti sono prefissati a circa il 20% in più del corpo a seconda delle classi dei documenti. Se proprio si desidera allargare o restringere l'avanzamento di riga rispetto al valore di default (a parte scriversi un file di classe personalizzato) si può usare il comando `\linespread` il cui argomento funziona da moltiplicatore dell'avanzamento di default. Per limitarne l'efficacia bisogna dare questo comando dentro un ambiente oppure dentro un gruppo. Si tenga però presente che l'effetto estetico combinato del corpo e dello scartamento, insieme all'effetto che questi valori hanno sulla comodità di lettura, sono estremamente delicati e solo un grafico editoriale sa come modificare i valori di default di frazioni di punto percentuale; se un inesperto si mette a giocare con `\linespread` sappia che non è consigliabile superare mai il 5% in più o in meno; valori maggiori producono di solito risultati pessimi.

	’00	’01	’02	’03	’04	’05	’06	’07	’10	’11	’12	’13	’14	’15	’16	’17	
’000	Γ 0	Δ 1	Θ 2	Λ 3	Ξ 4	Π 5	Σ 6	Υ 7	Φ 8	Ψ 9	Ω 10	ff 11	fi 12	fl 13	ffi 14	ffl 15	"00
’020	ı 16	ȷ 17	` 18	´ 19	˘ 20	¯ 21	˚ 22	¸ 23	ß 24	æ 25	œ 26	ø 27	Æ 28	Œ 29	Ø 30	/ 31	"10
’040	␣ 32	! 33	” 34	# 35	$ 36	% 37	& 38	’ 39	(40	) 41	* 42	+ 43	, 44	- 45	. 46	/ 47	"20
’060	0 48	1 49	2 50	3 51	4 52	5 53	6 54	7 55	8 56	9 57	: 58	; 59	¡ 60	= 61	¿ 62	? 63	"30
’100	@ 64	A 65	B 66	C 67	D 68	E 69	F 70	G 71	H 72	I 73	J 74	K 75	L 76	M 77	N 78	O 79	"40
’120	P 80	Q 81	R 82	S 83	T 84	U 85	V 86	W 87	X 88	Y 89	Z 90	[91	“ 92	] 93	^ 94	˙ 95	"50
’140	‘ 96	a 97	b 98	c 99	d 100	e 101	f 102	g 103	h 104	i 105	j 106	k 107	l 108	m 109	n 110	o 111	"60
’160	p 112	q 113	r 114	s 115	t 116	u 117	v 118	w 119	x 120	y 121	z 122	– 123	— 124	˝ 125	~ 126	¨ 127	"70
	"00	"01	"02	"03	"04	"05	"06	"07	"08	"09	"0A	"0B	"0C	"0D	"0E	"0F	

Parametri

inclinazione	0,00000	x-height	4,30554 pt
spazio interparola	3,33332 pt	larghezza del quadrato	10,19999 pt
allungamento interparola	1,66665 pt	spazio extra	1,11111 pt
accorciamento interparola	1,11111 pt	corpo nominale	10,00000 pt

Tabella 18.3: Il font latino a 128 caratteri con codifica OT1

	’00	’01	’02	’03	’04	’05	’06	’07	’10	’11	’12	’13	’14	’15	’16	’17	
’000	Њ 0	Љ 1	Ц 2	Э 3	І 4	Є 5	Ъ 6	Ћ 7	њ 8	љ 9	ц 10	э 11	і 12	є 13	ђ 14	ћ 15	"00
’020	Ю 16	Ж 17	Й 18	Ё 19	Ѵ 20	Ѳ 21	S 22	Я 23	ю 24	ж 25	й 26	ё 27	ѵ 28	ѳ 29	s 30	я 31	"10
’040	¨ 32	! 33	” 34	Ѣ 35	˘ 36	% 37	´ 38	’ 39	(40	) 41	* 42	Ь 43	, 44	- 45	. 46	/ 47	"20
’060	0 48	1 49	2 50	3 51	4 52	5 53	6 54	7 55	8 56	9 57	: 58	; 59	« 60	ı 61	» 62	? 63	"30
’100	ˇ 64	А 65	Б 66	Ц 67	Д 68	Е 69	Ф 70	Г 71	Х 72	И 73	Ј 74	К 75	Л 76	М 77	Н 78	О 79	"40
’120	П 80	Ч 81	Р 82	С 83	Т 84	У 85	В 86	Ш 87	Щ 88	Ы 89	З 90	[91	“ 92	] 93	Ь 94	Ъ 95	"50
’140	‘ 96	а 97	б 98	ц 99	д 100	е 101	ф 102	г 103	х 104	и 105	ј 106	к 107	л 108	м 109	н 110	о 111	"60
’160	п 112	ч 113	р 114	с 115	т 116	у 117	в 118	ш 119	щ 120	ы 121	з 122	– 123	— 124	№ 125	ь 126	ъ 127	"70
	"00	"01	"02	"03	"04	"05	"06	"07	"08	"09	"0A	"0B	"0C	"0D	"0E	"0F	

Parametri

inclinazione	0,00000	x-height	4,30554 pt
spazio interparola	3,77774 pt	larghezza del quadrato	11,55450 pt
allungamento interparola	1,74996 pt	spazio extra	1,16665 pt
accorciamento interparola	1,16665 pt	corpo nominale	10,00000 pt

Tabella 18.4: Il font cirillico a 128 caratteri con codifica OT2

18.3.2 La scelta delle altre caratteristiche

Le altre caratteristiche dei font, tranne la codifica, che, come si è detto, non viene cambiata quasi mai o viene cambiata automaticamente quando si cambia alfabeto, possono essere modificate autonomamente l’una dall’altra con i comandi già descritti; se una certa combinazione di famiglia, serie e forma non è disponibile, LaTeX provvede da solo a sostituire il font mancante con un font di default o con un font che abbia il maggior numero possibile delle caratteristiche richieste; nella tabella 18.6 vengono illustrate quasi tutte le combinazioni. Si noterà che in

	’00	’01	’02	’03	’04	’05	’06	’07	’10	’11	’12	’13	’14	’15	’16	’17	
’000	` 0	´ 1	^ 2	~ 3	¨ 4	˝ 5	° 6	ˇ 7	˘ 8	¯ 9	· 10	¸ 11	˛ 12	, 13	‹ 14	› 15	"00
’020	“ 16	” 17	„ 18	« 19	» 20	– 21	— 22	23	ı 24	ȷ 25	ff 26	fi 27	fl 28	ffi 29	ffl 30	31	"10
’040	␣ 32	! 33	" 34	# 35	$ 36	% 37	& 38	’ 39	(40	) 41	* 42	+ 43	, 44	- 45	. 46	/ 47	"20
’060	0 48	1 49	2 50	3 51	4 52	5 53	6 54	7 55	8 56	9 57	: 58	; 59	< 60	= 61	> 62	? 63	"30
’100	@ 64	A 65	B 66	C 67	D 68	E 69	F 70	G 71	H 72	I 73	J 74	K 75	L 76	M 77	N 78	O 79	"40
’120	P 80	Q 81	R 82	S 83	T 84	U 85	V 86	W 87	X 88	Y 89	Z 90	[91	\ 92	] 93	^ 94	_ 95	"50
’140	‘ 96	a 97	b 98	c 99	d 100	e 101	f 102	g 103	h 104	i 105	j 106	k 107	l 108	m 109	n 110	o 111	"60
’160	p 112	q 113	r 114	s 115	t 116	u 117	v 118	w 119	x 120	y 121	z 122	{ 123	| 124	} 125	~ 126	- 127	"70
’200	Ă 128	Ą 129	Ć 130	Č 131	Ď 132	Ě 133	Ę 134	Ğ 135	Ĺ 136	Ľ 137	Ł 138	Ń 139	Ň 140	Ŋ 141	Ő 142	Ŕ 143	"80
’220	Ř 144	Ś 145	Š 146	Ş 147	Ť 148	Ţ 149	Ű 150	Ů 151	Ÿ 152	Ź 153	Ž 154	Ż 155	IJ 156	İ 157	đ 158	§ 159	"90
’240	ă 160	ą 161	ć 162	č 163	ď 164	ě 165	ę 166	ğ 167	ĺ 168	ľ 169	ł 170	ń 171	ň 172	ŋ 173	ő 174	ŕ 175	"A0
’260	ř 176	ś 177	š 178	ş 179	ť 180	ţ 181	ű 182	ů 183	ÿ 184	ź 185	ž 186	ż 187	ij 188	¡ 189	¿ 190	£ 191	"B0
’300	À 192	Á 193	Â 194	Ã 195	Ä 196	Å 197	Æ 198	Ç 199	È 200	É 201	Ê 202	Ë 203	Ì 204	Í 205	Î 206	Ï 207	"C0
’320	Đ 208	Ñ 209	Ò 210	Ó 211	Ô 212	Õ 213	Ö 214	Œ 215	Ø 216	Ù 217	Ú 218	Û 219	Ü 220	Ý 221	Þ 222	SS 223	"D0
’340	à 224	á 225	â 226	ã 227	ä 228	å 229	æ 230	ç 231	è 232	é 233	ê 234	ë 235	ì 236	í 237	î 238	ï 239	"E0
’360	ð 240	ñ 241	ò 242	ó 243	ô 244	õ 245	ö 246	œ 247	ø 248	ù 249	ú 250	û 251	ü 252	ý 253	þ 254	ß 255	"F0
	"00	"01	"02	"03	"04	"05	"06	"07	"08	"09	"0A	"0B	"0C	"0D	"0E	"0F	

Parametri

inclinazione	0,00000	x-height	4,30450 pt
spazio interparola	3,33251 pt	larghezza del quadrato	9,99755 pt
allungamento interparola	1,66625 pt	spazio extra	1,11083 pt
accorciamento interparola	1,11083 pt	corpo nominale	10,00000 pt

Tabella 18.5: Il font latino a 256 caratteri con codifica T1

alcune caselle il contenuto è rosso; LaTeX ha sostituito il font mancante con un font che abbia un certo numero di caratteristiche fra quelle richieste.

Per confrontare l’effetto dei comandi per la scelta dei font in modo testo o in modo matematico si esamini il medaglione nella pagina 738.

Vale la pena di mettere in guardia gli utenti che si servono di modelli di documento composti per il vecchio LaTeX 209, nei quali si usano comandi abbreviati del tipo `\rm` per scegliere il font tondo, o `\it` per scegliere il corsivo, eccetera. Quei vecchi comandi possono essere stati conservati per compatibilità in alcune classi standard, ma operano molto diversamente dai comandi della tabella 18.6. Essi infatti simulano perfettamente anche in LaTeX 2_ε il funzionamento che essi avevano in LaTeX 209. Cioè essi impostano una sola caratteristica ripristinando le altre caratteristiche ai valori di default; essi, di fatto, eseguono un comando `\normalfont` che imposta tutti i valori di default (famiglia con grazie, forma diritta, serie media) e poi specificano solo la famiglia o solo la forma o solo la serie richiesta. È ovvio che quei vecchi comandi *non* devono venire più usati.

Inoltre l’abitudine all’uso di quei vecchi comandi può indurre a pensare che la dichiarazione `\rmfamily` e il corrispondente comando `\textrm` siano in tutto e per tutto equivalenti a `\rm`: evidentemente questo è falso, perché la prima

		\upshape	\slshape	\itshape	\scshape	\uishape
\mdseries	\rmfamily	abcABC	*abcABC*	*abcABC*	ABCABC	abcABC
	\sffamily	abcABC	*abcABC*	*abcABC*	ABCABC	abcABC
	\ttfamily	abcABC	*abcABC*	*abcABC*	ABCABC	abcABC
\bfseries	\rmfamily	**abcABC**	***abcABC***	***abcABC***	**abcABC**	**abcABC**
	\sffamily	**abcABC**	***abcABC***	***abcABC***	**abcABC**	**abcABC**
	\ttfamily	**abcABC**	***abcABC***	***abcABC***	**abcABC**	**abcABC**

Tabella 18.6: Le varie combinazioni di serie e di forma per le varie famiglie standard dei font Computer Modern usabili con LaTeX. I caratteri rossi sono stati sostituiti con altri caratteri aventi il maggior numero possibile delle caratteristiche richieste. Con i font latini aventi la codifica T1 si hanno meno sostituzioni; solo la famiglia *typewriter type* continua a non disporre della serie nera. La collezione dei font latini Latin Modern oltre la famiglia *typewriter type*, contiene anche la famiglia *variable typewriter type* (macchina da scrivere a spaziatura variabile); per entrambe esiste la serie nera ma solo nelle forme dritta e inclinata.

dichiarazione e il corrispondente comando impostano solo la famiglia, lasciando inalterata la forma e la serie preesistenti; `\rm`, invece, reimposta tutti i valori di default, anche la serie e la forma, e poi specifica la forma diritta. Gli utenti dei vecchi comandi di LaTeX 209, sono quindi pregati di dimenticarli completamente, benché avessero qualche funzionalità comoda, come per esempio quella di poter essere usati anche in matematica.

Piuttosto tutti gli utenti ricordino che i nuovi comandi `\text`$\langle xx \rangle$ possono venire usati anche in matematica e, sotto certe condizioni (vedi più avanti), persino in posizioni di apici o di pedici, per scrivere brevi apposizioni testuali, dove però gli spazi, gli accenti testuali, eccetera possono essere usati regolarmente come se si trattasse di comporre del testo normale. Benché sia possibile, nel paragrafo 18.5 verrà spiegato in dettaglio perché non sia opportuno servirsi di questi comandi per i pedici e per gli apici.

Nella tabella 18.6 compaiono solo le istruzioni per assegnare una certa caratteristica al font scelto. Fra le altre cose si è indicata l'istruzione `\uishape` che in LaTeX non esiste; la si è definita mediante un comando, che verrà descritto meglio in seguito, in modo da creare un comando per l'utente che non lo obblighi a ricorrere a istruzioni di livello inferiore:

```
\newcommand{\uishape}{\fontshape{ui}\selectfont}
\DeclareTextFontCommand{\textui}{\uishape}
```

Le dichiarazioni usate nella composizione della tabella possono essere usate anche durante la composizione del testo; tuttavia per comporre singole parole o brevi frasi è preferibile ricorrere ai comandi; tutti sono della forma `\text`$\langle xx \rangle$, dove $\langle xx \rangle$ sono le stesse due lettere che compaiono all'inizio delle istruzioni; avremo quindi `\textrm`, `\textsl`, `\textbf`, `\textsf`, eccetera.

Attenzione: Nel 2020 il LaTeX 3 Project Team ha esteso i comandi di serie e di forma; la cosa è diventata più delicata, ma mentre la tabella 18.6 mostra i comandi abitualmente usati dalla maggioranza degli utenti, la guida dei font [1] descrive la moltitudine di nuovi codici da usare in circostanze molto particolari, nei dettagli delle quali qui non è opportuno scendere; si veda anche quanto scritto a questo proposito nel capitolo 30.

La sintassi di tutti questi comandi è la seguente:

```
\text⟨xx⟩{⟨testo⟩}
```

Perché è preferibile usare i comandi rispetto alle istruzioni? Perché con il comando il programma LaTeX può conoscere dove comincia e dove finisce il testo da comporre con una particolare caratteristica e quindi sa dove inserire la 'correzione italica', una piccola correzione per allontanare o per avvicinare i segni che precedono o seguono il ⟨*testo*⟩. Spesso i caratteri che seguono sono dei segni di interpunzione 'alti' come i due punti o il punto e virgola, oppure come il punto esclamativo o interrogativo; talvolta si tratta di parentesi; si noti allora l'effetto di chiudere fra parentesi una parola scritta in corsivo se non è presente o se è presente la correzione italica: (*XYZ*) a confronto con (*XYZ*). Chiaramente il secondo caso è stato composto con il comando e il primo con l'istruzione; nel primo la parentesi chiusa batte contro la *Z* nel secondo no[4]. Si osservi infine che il caso più frequente di necessità della correzione italica si ha quando si usa il carattere corsivo; tuttavia il fenomeno si produce anche con il tondo[5]: fF e fF.

Fra i comandi per scegliere la forma dei segni di un font non bisogna scordare l'istruzione `\em` e il comando `\emph`; essi servono per evidenziare (in inglese *emphasize*) una parola o una breve locuzione all'interno di altro testo. Questa istruzione e questo comando hanno la proprietà di cambiare inclinazione al font (eventualmente passando dal tondo al corsivo, oppure al tondo inclinato), per cui in questo testo scritto in tondo la parola *emphasize* è evidenziata in corsivo, mentre *in questo testo composto in corsivo* emphasize *viene evidenziato in tondo*. In entrambi i casi si è scritto `\emph{emphasize}`.

È d'obbligo segnalare un fatto importante; le varie forme, serie e corpi, nonché le varie famiglie di font vanno usate con criterio. L'abitudine di evidenziare le parole in corsivo o in neretto, oppure ancora in corsivo nero è una operazione da inesperto. Esistono delle norme editoriali per l'uso di queste varianti dei caratteri e non è il caso di abusarne. In tipografia non si sottolineano mai parole o frasi per metterle in evidenza, come si faceva con la macchina da scrivere. Esistono poi anche delle 'norme' psicosociologiche che sconsigliano di cambiare font troppo spesso; se qualcosa deve essere evidenziato, questo non vuol dire evidenziare tutto. Alcuni studenti usano il pennarello colorato per evidenziare le parti che ritengono importanti; finiscono con l'evidenziare quasi tutto e il risultato ovvio è che nulla emerge dallo sfondo colorato, se non, appunto, le cose meno importanti!

[4] Si noti che in tipografia si usano le parentesi non inclinate anche quando esse racchiudono testo scritto con caratteri inclinati.

[5] L'esempio non è cervellotico: fF è il simbolo dell'unità di misura 'femtofarad' che si incontra spesso in microelettronica.

18.4 Altri font diversi da quelli di default

Talvolta è necessario usare font diversi da quelli di default per ottenere effetti diversi, per uniformarsi allo stile di documenti scritti con altri metodi, per aumentare la leggibilità, per rendere più compatto il testo, eccetera.

Sono stati predisposti diversi pacchetti per sostituire le famiglie di font usate di default con altre famiglie, generalmente con lo scopo di usare famiglie di font commerciali, anche se disponibili gratuitamente. Sul mercato, infatti, sono disponibili migliaia di famiglie di font. Essi costituiscono la croce e delizia dei disegnatori editoriali e richiedono, appunto, la loro professionalità per usarli al meglio. Per gli inesperti può essere difficile da un punto di vista estetico e funzionale usare i font commerciali, ma la presenza dei pacchetti per l'uso di questi font rende l'operazione tecnicamente molto più facile e l'uso di questi font diventa alla portata di tutti.

Si possono dunque invocare diversi pacchetti che servono per sostituire una alla volta le tre famiglie principali di font; tuttavia la cosa non è semplice, o meglio è semplicissimo con LaTeX, ma è difficile eseguire scelte in cui le famiglie si accordino alla perfezione, in particolare per quel che riguarda l'altezza delle minuscole, la famosa *x-height*.

Per questo scopo, allora si consiglia ai lettori di fare riferimento a due pacchetti in particolare: *txfonts* e *pxfonts*. Meglio ancora: recentemente sono stati caricati su CTAN e inseriti nella distribuzione di TeX Live, le revisioni di questi due pacchetti; la documentazione si legge con `texdoc newtxfonts` e `texdoc newpxfonts`, ma i pacchetti da caricare sono distinti per il testo e per la matematica: quindi si dispone di *newtxtext* e *newtxmath* per i nuovi font Times, e *newpxtext* e *newpxmath* per i Palatino. Questa separazione consente di usare i font per la matematica anche con altri font testuali, e viceversa di usare altri font matematici con i nuovi Times o Palatino. Questa operazione può portare ad abbinamenti non tanto buoni, quindi è opportuno leggere con attenzione le documentazioni indicate sopra, dove viene spiegato più dettagliatamente il perché di questa novità.

Il lettore confronti i capoversi presenti nella figura 18.6; si tratta di due testi composti con font dello stesso corpo (10 pt) e con lo stesso scartamento (12 pt), ma con collezioni diverse di font. Eventualmente usi una lente di ingrandimento (anche quella fornita dal programma di visualizzazione del file PDF) per esaminare i dettagli di forma delle lettere; inoltre noti il differente livello di nero dei testi composti con questi font.

18.5 I font per la matematica

I font per la matematica sono un argomento un po' particolare perché sono una specialità del sistema TeX. Ci sono anche altri font dotati di segni matematici, ma di solito sono assai modesti come qualità tipografica, a meno che non siano stati adattati completamente al sistema TeX, come per esempio i font Times

Testo composto con i nuovi font Times extended

Il pacchetto **newtxtext** consente di usare i font Times per il testo con grazie, il font Helvetica per il testo scritto senza grazie, e una variante del Courier per la famiglia a spaziatura fissa. I font sono caricati con fattori di scala predefiniti in modo che le altezze *x-height* siano il più possibile simili, non esattamente identiche, perché la forma dei segni richiede delle curve leggermente abbondanti specialmente con l'Helvetica.

La 'x' presente nel nome di questo pacchetto richiama il fatto che si tratta di una collezione estesa; infatti con l'uso di questi font diventano disponibili non solo tutti i simboli di LaTeX standard arricchito dai font del pacchetto **textcomp** ma una ulteriore varietà che ne aumenta ancora la versatilità. Quindi **tx** sta per 'Times eXtended'.

Testo composto con i nuovi font Palatino extended

L'altro pacchetto **newpxtext** mette a disposizione il Palatino esteso; il Palatino viene usato al posto del Times, ma per le altre famiglie si usano ancora l'Helvetica e il Courier. In questo caso i fattori di scala con cui sono invocati i font accessori sono diversi da quelli del Times, perché il Palatino a pari corpo appare decisamente più grande e più largo.

Figura 18.6: Esempi di testi composti con i nuovi font Times e Palatino estesi

forniti con il pacchetto `txfonts` o `newtxmath`, oppure i font Palatino forniti con il pacchetto `pxfonts` o `newpxmath`; di questi font si è già detto in precedenza.

Tuttavia è doveroso presentare anche le tabelle dei font matematici, affinché, sia pure sotto forma di una specie di catalogo, ci si possa rendere conto di quali segni sono disponibili e quali sono gli indirizzi di ogni segno all'interno di ogni polizza.

Le tabelle 18.7 e 18.8 contengono la polizza del corsivo matematico, che ha caratteristiche assai diverse dal font corsivo per il testo; la polizza contiene anche un certo numero di simboli e l'alfabeto greco maiuscolo e minuscolo oltre alle cifre arabe *minuscole*. I grandi maestri della tipografia unanimemente raccomandano di usare le cifre arabe minuscole nel testo e di usare quelle maiuscole nelle tabelle e in matematica. La polizza dei simboli, oltre a numerosi simboli matematici e per altri scopi, contiene anche l'alfabeto calligrafico maiuscolo; l'alfabeto calligrafico viene selezionato in matematica con il comando `\mathcal`.

Vale la pena di commentare la polizza del segni estensibili 18.9, perché la tabella sembra formata in modo diverso dalle altre. Non è 'colpa' della tabella, ma dei font; tutti i segni hanno una altezza nulla e una profondità corrispondente alla effettiva dimensione verticale del segno; questo serve alla sezione di composizione della matematica per poter collocare correttamente tutti i segni in quella disposizione bidimensionale costituita da una espressione matematica. Si notino anche i mezzi delimitatori e i pezzetti che permettono, sovrapponendoli

	’00	’01	’02	’03	’04	’05	’06	’07	’10	’11	’12	’13	’14	’15	’16	’17	
’000	Γ_{0}	Δ_{1}	Θ_{2}	Λ_{3}	Ξ_{4}	Π_{5}	Σ_{6}	Υ_{7}	Φ_{8}	Ψ_{9}	Ω_{10}	α_{11}	β_{12}	γ_{13}	δ_{14}	ϵ_{15}	"00
’020	ζ_{16}	η_{17}	θ_{18}	ι_{19}	κ_{20}	λ_{21}	μ_{22}	ν_{23}	ξ_{24}	π_{25}	ρ_{26}	σ_{27}	τ_{28}	υ_{29}	ϕ_{30}	χ_{31}	"10
’040	ψ_{32}	ω_{33}	ε_{34}	ϑ_{35}	ϖ_{36}	ϱ_{37}	ς_{38}	φ_{39}	$\leftharpoonup_{40}$	$\leftharpoondown_{41}$	$\rightharpoonup_{42}$	$\rightharpoondown_{43}$	$\triangleleft_{44}$	$\triangleright_{45}$	$\triangleright_{46}$	$\triangleleft_{47}$	"20
’060	0_{48}	1_{49}	2_{50}	3_{51}	4_{52}	5_{53}	6_{54}	7_{55}	8_{56}	9_{57}	$.{}_{58}$	$,{}_{59}$	$<_{60}$	$/_{61}$	$>_{62}$	$\star_{63}$	"30
’100	∂_{64}	A_{65}	B_{66}	C_{67}	D_{68}	E_{69}	F_{70}	G_{71}	H_{72}	I_{73}	J_{74}	K_{75}	L_{76}	M_{77}	N_{78}	O_{79}	"40
’120	P_{80}	Q_{81}	R_{82}	S_{83}	T_{84}	U_{85}	V_{86}	W_{87}	X_{88}	Y_{89}	Z_{90}	$\flat_{91}$	$\natural_{92}$	$\sharp_{93}$	$\smile_{94}$	$\frown_{95}$	"50
’140	ℓ_{96}	a_{97}	b_{98}	c_{99}	d_{100}	e_{101}	f_{102}	g_{103}	h_{104}	i_{105}	j_{106}	k_{107}	l_{108}	m_{109}	n_{110}	o_{111}	"60
’160	p_{112}	q_{113}	r_{114}	s_{115}	t_{116}	u_{117}	v_{118}	w_{119}	x_{120}	y_{121}	z_{122}	$\imath_{123}$	$\jmath_{124}$	$\wp_{125}$	$\rightarrow_{126}$	$\mathbin{\frown}_{127}$	"70
	"00	"01	"02	"03	"04	"05	"06	"07	"08	"09	"0A	"0B	"0C	"0D	"0E	"0F	

Parametri

inclinazione	0,25000	x-height	4,30554 pt
spazio interparola	0,00000 pt	larghezza del quadrato	10,19999 pt
allungamento interparola	0,00000 pt	spazio extra	0,00000 pt
accorciamento interparola	0,00000 pt	corpo nominale	10,00000 pt

Tabella 18.7: Il corsivo matematico con codifica OML; questa polizza contiene anche le cifre arabe ‘minuscole’.

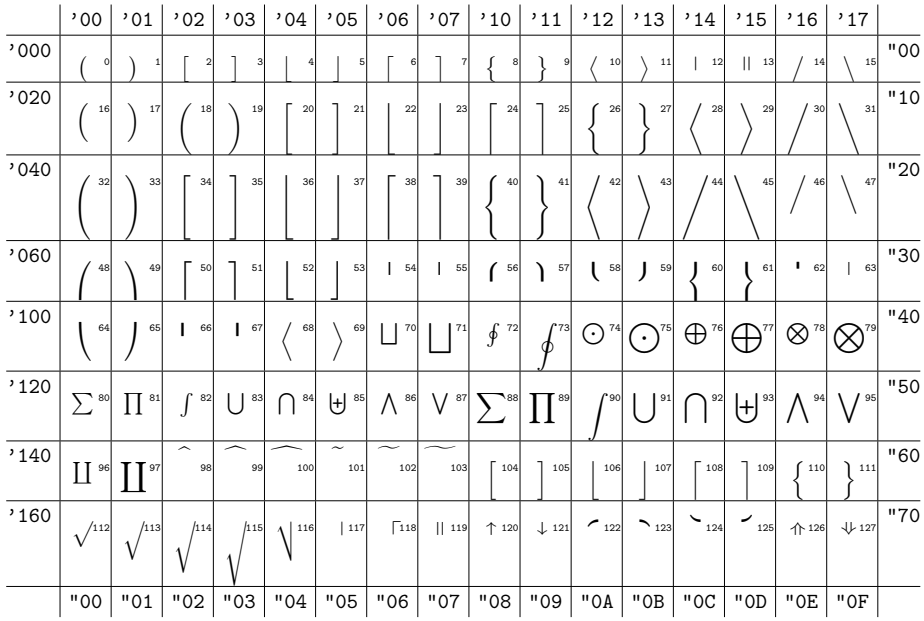
	’00	’01	’02	’03	’04	’05	’06	’07	’10	’11	’12	’13	’14	’15	’16	’17	
’000	$-_{0}$	$\cdot_{1}$	$\times_{2}$	$*_{3}$	$\div_{4}$	$\diamond_{5}$	$\pm_{6}$	$\mp_{7}$	$\oplus_{8}$	$\ominus_{9}$	$\otimes_{10}$	$\oslash_{11}$	$\odot_{12}$	$\bigcirc_{13}$	$\circ_{14}$	$\bullet_{15}$	"00
’020	$\asymp_{16}$	$\equiv_{17}$	$\subseteq_{18}$	$\supseteq_{19}$	$\leq_{20}$	$\geq_{21}$	$\preceq_{22}$	$\succeq_{23}$	$\sim_{24}$	$\approx_{25}$	$\subset_{26}$	$\supset_{27}$	$\ll_{28}$	$\gg_{29}$	$\prec_{30}$	$\succ_{31}$	"10
’040	$\leftarrow_{32}$	$\rightarrow_{33}$	$\uparrow_{34}$	$\downarrow_{35}$	$\leftrightarrow_{36}$	$\nearrow_{37}$	$\searrow_{38}$	$\simeq_{39}$	$\Leftarrow_{40}$	$\Rightarrow_{41}$	$\Uparrow_{42}$	$\Downarrow_{43}$	$\Leftrightarrow_{44}$	$\nwarrow_{45}$	$\swarrow_{46}$	$\propto_{47}$	"20
’060	$\prime_{48}$	∞_{49}	$\in_{50}$	$\ni_{51}$	$\triangle_{52}$	$\triangledown_{53}$	$/_{54}$	$\mathbin{.}_{55}$	$\forall_{56}$	$\exists_{57}$	$\neg_{58}$	$\emptyset_{59}$	$\Re_{60}$	$\Im_{61}$	$\top_{62}$	$\bot_{63}$	"30
’100	$\aleph_{64}$	$\mathcal{A}_{65}$	$\mathcal{B}_{66}$	$\mathcal{C}_{67}$	$\mathcal{D}_{68}$	$\mathcal{E}_{69}$	$\mathcal{F}_{70}$	$\mathcal{G}_{71}$	$\mathcal{H}_{72}$	$\mathcal{I}_{73}$	$\mathcal{J}_{74}$	$\mathcal{K}_{75}$	$\mathcal{L}_{76}$	$\mathcal{M}_{77}$	$\mathcal{N}_{78}$	$\mathcal{O}_{79}$	"40
’120	$\mathcal{P}_{80}$	$\mathcal{Q}_{81}$	$\mathcal{R}_{82}$	$\mathcal{S}_{83}$	$\mathcal{T}_{84}$	$\mathcal{U}_{85}$	$\mathcal{V}_{86}$	$\mathcal{W}_{87}$	$\mathcal{X}_{88}$	$\mathcal{Y}_{89}$	$\mathcal{Z}_{90}$	$\cup_{91}$	$\cap_{92}$	$\uplus_{93}$	$\wedge_{94}$	$\vee_{95}$	"50
’140	$\vdash_{96}$	$\dashv_{97}$	$\lfloor_{98}$	$\rfloor_{99}$	$\lceil_{100}$	$\rceil_{101}$	$\{_{102}$	$\}_{103}$	$\langle_{104}$	$\rangle_{105}$	$\vert_{106}$	$\Vert_{107}$	$\updownarrow_{108}$	$\Updownarrow_{109}$	$\setminus_{110}$	$\wr_{111}$	"60
’160	$\surd_{112}$	$\amalg_{113}$	∇_{114}	$\int_{115}$	$\sqcup_{116}$	$\sqcap_{117}$	$\sqsubseteq_{118}$	$\sqsupseteq_{119}$	$\S_{120}$	$\dagger_{121}$	$\ddagger_{122}$	$\P_{123}$	$\clubsuit_{124}$	$\diamondsuit_{125}$	$\heartsuit_{126}$	$\spadesuit_{127}$	"70
	"00	"01	"02	"03	"04	"05	"06	"07	"08	"09	"0A	"0B	"0C	"0D	"0E	"0F	

Parametri

inclinazione	0,25000	x-height	4,30554 pt
spazio interparola	0,00000 pt	larghezza del quadrato	10,19999 pt
allungamento interparola	0,00000 pt	spazio extra	0,00000 pt
accorciamento interparola	0,00000 pt	corpo nominale	10,00000 pt

Tabella 18.8: La polizza dei simboli matematici non estensibili con codifica OMS. Questa polizza è usata anche come alfabeto matematico, perché contiene anche le maiuscole calligrafiche.

in modo scalato, di disegnare qualunque delimitatore con qualsiasi dimensione sia in altezza per quelli che devono espandersi in verticale, sia in larghezza per quelli che devono espandersi in orizzontale.



Parametri			
inclinazione	0,00000	x-height	4,30554 pt
spazio interparola	0,00000 pt	larghezza del quadrato	10,19999 pt
allungamento interparola	0,00000 pt	spazio extra	0,00000 pt
accorciamento interparola	0,00000 pt	corpo nominale	10,00000 pt

Tabella 18.9: I segni matematici estensibili con codifica OMX

La gestione di questi simboli estensibili, come si può facilmente immaginare, è piuttosto complessa; il bello è che l'interprete fa tutto da solo e il compositore non deve occuparsi di nulla per gestire questi simboli; al massimo deve ricordarsi di indicare con `\left` i delimitatori di apertura e con `\right` i delimitatori di chiusura. Usando pdftex (che, ricordiamo, include anche le estensioni di etex) come interprete, esiste anche il comando `\middle` per un delimitatore intermedio che viene esteso in verticale nello stesso modo di quando si usa `\left` o `\right` ma con una spaziatura a destra e a sinistra coerente con il significato di delimitatore intermedio.

Solo lualatex e xelatex

Per i programmi xelatex e lualatex i font matematici si impostano caricando il file di estensione *unicode-math* e caricando una famiglia di font matematici OpenType. Il sistema TEX viene distribuito con una certa collezione di questi font matematici OpenType i cui nomi contengono tutti il nome di un font testuale con

il suffisso ␣`Math`. Così si avrà la famiglia XITS Math, Asana Math, Latin Modern Math, eccetera: il sistema TeX e il sistema operativo mettono a disposizione un'altra quindicina di font OpenType matematici. Le impostazioni nel preambolo sono semplicemente le seguenti:

```
\usepackage{amsmath} % quando è necessario
\usepackage{unicode-math}
\setmathfont{⟨famiglia di font matematici OpenType⟩}
```

Si noti che se si vuole usare il file di estensione *amsmath* lo si deve fare prima di caricare *unicode-math*, ma *non si deve mai caricare* il file *amssymb*, perché tutti i simboli matematici sono già forniti dal file *unicode-math* e dal font matematico prescelto. Usando questi programmi l'uso dei font matematici è molto più semplice rispetto a quando si usa pdflatex; quanto descritto qui di seguito riguarda solo quest'ultimo programma.

Solo pdflatex

In matematica la scelta dei font è molto più complessa. Innanzi tutto è possibile, di default, scrivere una formula in caratteri neri, solo se si segue una delle regole seguenti.

1. Prima di entrare in modo matematico si specifica la dichiarazione `\boldmath`; chiuso l'ambiente matematico bisogna poi specificare il comando `\unboldmath`, altrimenti la dichiarazione di comporre in nero rimane valida 'per sempre'; in alternativa la formula da comporre in nero può essere racchiusa, assieme alla dichiarazione `\boldmath`, dentro ad un gruppo, per esempio:

   ```
   {% inizio del gruppo
   \boldmath
   \begin{equation}
   ...
   \end{equation}
   }% fine del gruppo
   ```

2. Si usa il pacchetto *amsmath* con il pacchetto accessorio *amssymb* che mette a disposizione il comando `\boldsymbol` per comporre in nero un simbolo matematico isolatamente; oppure si carica il pacchetto *bm* che mette a disposizione un comando simile, `\bm`, per comporre in nero simboli o sottoespressioni matematiche isolate o, volendo, una intera espressione.

I comandi disponibili fanno riferimento agli alfabeti matematici caricati dalla classe o dai pacchetti richiamati nel preambolo; di default pdflatex carica i font della collezione Computer Modern a 128 caratteri predisposti per la matematica, tabelle 18.7-18.9; questi font obbediscono in generale a codifiche specifiche non usabili nel testo se non con comandi particolari. Per esempio è possibile usare nel testo le cifre arabe *minuscole* 0123456789 attraverso l'uso del comando `\oldstylenums`, sebbene i loro segni derivino dal font destinato alle lettere matematiche.

In particolare ogni classe e/o ogni pacchetto che carichi i font matematici di default, o alternativi a quelli di default, carica quattro famiglie[6] distinte.

1. La famiglia per gli operatori (tabella 18.3); essa può coincidere con il font tondo con grazie usato per il testo, ma viene usato in modo matematico: non accetta gli accenti testuali e non accetta gli spazi. Esso serve per comporre i più comuni simboli matematici come $+, -, =$ e pochi altri, le cifre arabe *maiuscole* diritte, ma essenzialmente serve per comporre il nome delle funzioni matematiche come 'sin', 'cos', 'log', 'lim', eccetera.
2. La famiglia per le lettere (tabella 18.7); essa contiene l'alfabeto latino e l'alfabeto greco corsivi usati per le variabili matematiche; contiene anche numerosi altri simboli binari o di relazione e le cifre arabe *minuscole*. Il corsivo matematico latino non riconosce le legature, perché ogni lettera rappresenta una variabile e in matematica le variabili moltiplicate fra di loro sono generalmente scritte una dopo l'altra senza segni interposti. Si può scrivere $ff = f^2$ e le due f non devono legarsi nella solita legatura testuale *ff*. Usare questi font per scrivere V_{eff} per indicare la tensione *efficace* produce una composizione orrenda. Si è già detto come scrivere correttamente pedici e/o apici testuali usati come apposizioni delle variabili matematiche.
3. La famiglia per i simboli (tabella 18.8); contiene l'alfabeto maiuscolo *calligrafico* e moltissimi altri segni per gli operatori binari e/o di relazione.
4. La famiglia dei simboli estensibili (tabella 18.9), cioè i segni di radice e di integrale, e le parentesi di vari tipi che cambiano dimensioni a seconda del contenuto su cui devono operare.

Siccome le famiglie di font matematici in senso stretto che pdflatex può usare sono al massimo sedici, non è il caso di caricare altre famiglie senza tenere conto di questo limite; quando si carica direttamente o indirettamente il pacchetto *amsfonts* vengono definite altre due famiglie di segni matematici (tabelle 14.1 e 14.2). È raro raggiungere il limite di 16 famiglie, ma a chi scrive è capitato.

Per inserire variabili composte con altri alfabeti, mediante i comandi `\mathsf` o `\mathtt` o `\mathit`, o simili, pdflatex ricorre a dei comandi che usano i font testuali in modo matematico (senza accenti e senza spazi) ma senza caricare effettivamente altre famiglie di font matematici. I dettagli sono abbastanza delicati, per cui si rimanda alla guida `fntguide.pdf` [1] nella cartella `.../doc/latex/base/`, che dedica una lunga sezione ai comandi per gestire i font matematici. Per vedere in azione quei comandi, si può esaminare il file `fontmath.ltx` nella cartella `.../tex/latex/base/` dove sono usati tutti gli speciali comandi per caricare i font matematici di default.

[6] Il nome *famiglia* qui è usato in modo un po' diverso da solito; per TEX una famiglia di font matematici è un gruppo di font con lo stesso disegno, ma usabili in tre corpi distinti, quello normale, quello per i pedici e gli apici di primo livello e quello per i pedici e gli apici di secondo livello. Questo è uno dei motivi per i quali la gestione dei font matematici è decisamente più elaborata di quella per i font testuali. Con i font matematici OpenType, non si parla nemmeno di famiglie matematiche o gruppi matematici, perché con i font OpenType matematici basta un solo font per contenere tutti i glifi che servono.

Il consiglio che si può dare in merito alla gestione dei caratteri matematici è quello di cercare di avere sempre i font testuali e quelli matematici che si accordino fra di loro armoniosamente; come minimo devono avere lo stesso occhio delle lettere minuscole e devono avere la stessa nerezza. I vecchi pacchetti *txfonts* e *pxfonts* e quelli nuovi *newtxtext* con *newtxmath* e *newpxtext* con *newpxmath* caricano correttamente sia i font testuali sia quelli per la matematica. Se invece, per esempio, si volesse usare il font Utopia per il testo, basterebbe usare il pacchetto *utopia*; questo pacchetto, però, imposta solo i font testuali, conservando i font di default per la matematica. Il risultato che si otterrebbe è pessimo: l'occhio dei font Utopia e la loro nerezza contrasta vistosamente con l'occhio e la nerezza dei font Computer Modern, per cui l'uso di quel pacchetto sarebbe accettabile solo per comporre un documento *privo* completamente di matematica.

Il pacchetto *mathdesign*, specificando l'opportuna opzione, permette invece di caricare font testuali e matematici che si accordano fra di loro armoniosamente. Dando il comando

```
\usepackage[utopia]{mathdesign}
```

si possono usare infatti i font Utopia anche per testi contenenti della matematica, perché i font matematici sono scelti in modo da accordarsi con i font per il testo.

Il pacchetto *mathdesign* accetta come opzioni *utopia* (Adobe Utopia), *charter* (Bitstream Charter), *garamond* (URW Garamond), e ne sono previsti alcuni altri. Bisogna usare qualche piccola cautela (per esempio, non si deve contemporaneamente caricare il pacchetto *amsmath* con i suoi font, se non altro perché *mathdesign* ha già ridefinito tutti i comandi) e per questo è necessario leggere con attenzione il file di documentazione `mathdesign-doc.pdf` in `.../doc/latex/mathdesign/`. Anche il pacchetto *fourier* consente di usare il font Utopia con i caratteri matematici congruenti; il file di documentazione `fourier-doc-en.pdf` è in `.../doc/fonts/fourier/`. Anche il pacchetto *kpfonts* permette di usare una ottima imitazione dei font Palladio testuale a cui vengono affiancati dei font matematici disegnati apposta. Il pacchetto *libertinus* permette di usare i font testuali e matematici di una variante Libertinus dei font Linux Libertine. Un pregio unico di questi font è che usa font matematici codificati a 8 bit, quindi con 256 segni in ogni polizza, per cui si possono usare tranquillamente più famiglie (sempre al massimo 16, ma di default se ne usano di meno per la matematica ordinaria) rispetto a quanto si può ottenere con i font matematici "normali".

Con alcune classi di default vengono usati font testuali e matematici che possono venire facilmente modificati e sostituiti con altri. In particolare talvolta viene usato il pacchetto *euler*; il file della documentazione `euler.pdf` si trova in `.../doc/latex/euler/` e va letto attentamente; fra l'altro vi si dice che questo tipo di font matematico si accorda abbastanza bene con diversi font vettoriali, ma bisogna fare attenzione sulla sequenza delle chiamate dei vari pacchetti. Il pacchetto accetta anche diverse opzioni che vanno usate con attenzione. Tuttavia, benché i singoli caratteri siano bellissimi e siano una imitazione (disegnata dal grande Hermann Zapf) dei segni che un matematico con bella calligrafia

scriverebbe sulla lavagna, essi sono sostanzialmente diritti e non hanno una forma corsiva o comunque inclinata; per questo motivo essi sono da preferire per comporre un documento di tipo prettamente matematico e non risultano adatti ad un documento scritto da un fisico o da un tecnologo. Naturalmente è una questione di gusti, ma è opportuno ricordare che le norme ISO per la scrittura della matematica per la fisica e le scienze sperimentali sono piuttosto cogenti e hanno una loro precisa ragion d'essere.

	’00	’01	’02	’03	’04	’05	’06	’07	’10	’11	’12	’13	’14	’15	’16	’17	
’000	` 0	´ 1	ˆ 2	˜ 3	¨ 4	˝ 5	° 6	ˇ 7	˘ 8	¯ 9	˙ 10	¸ 11	˛ 12	’ 13	14	15	"00
’020	16	17	" 18	19	20	— 21	— 22	23	← 24	→ 25	⁀ 26	⁀ 27	⁀ 28	⁀ 29	30	31	"10
’040	ƀ 32	33	34	35	$ 36	37	38	’ 39	40	41	* 42	43	, 44	= 45	. 46	/ 47	"20
’060	0 48	1 49	2 50	3 51	4 52	5 53	6 54	7 55	8 56	9 57	58	59	⟨ 60	– 61	⟩ 62	63	"30
’100	64	65	66	67	68	69	70	71	72	73	74	75	76	℧ 77	78	◯ 79	"40
’120	80	81	82	83	84	85	86	Ω 87	88	89	90	⟦ 91	92	⟧ 93	↑ 94	↓ 95	"50
’140	` 96	97	⋆ 98	o|o 99	† 100	101	102	103	104	105	106	107	108	∞ 109	♩ 110	111	"60
’160	112	113	114	115	116	117	118	119	120	121	122	123	124	125	~ 126	= 127	"70
’200	◡ 128	◡ 129	″ 130	‶ 131	† 132	‡ 133	‖ 134	‰ 135	• 136	℃ 137	$ 138	¢ 139	f 140	₡ 141	₩ 142	₦ 143	"80
’220	₲ 144	₱ 145	£ 146	₨ 147	₹ 148	¡ 149	₫ 150	™ 151	‰ 152	¶ 153	฿ 154	№ 155	℅ 156	℮ 157	○ 158	℠ 159	"90
’240	{ 160	} 161	¢ 162	£ 163	¤ 164	¥ 165	¦ 166	§ 167	¨ 168	Ⓒ 169	ª 170	ⓑ 171	¬ 172	Ⓟ 173	Ⓡ 174	¯ 175	"A0
’260	° 176	± 177	² 178	³ 179	´ 180	µ 181	¶ 182	· 183	※ 184	¹ 185	º 186	√ 187	¼ 188	½ 189	¾ 190	€ 191	"B0
’300	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207	"C0
’320	208	209	210	211	212	213	× 214	215	216	217	218	219	220	221	222	223	"D0
’340	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239	"E0
’360	240	241	242	243	244	245	÷ 246	247	248	249	250	251	252	253	254	255	"F0
	"00	"01	"02	"03	"04	"05	"06	"07	"08	"09	"0A	"0B	"0C	"0D	"0E	"0F	

Parametri

inclinazione	0,00000	x-height	4,30450 pt
spazio interparola	3,33251 pt	larghezza del quadrato	9,99755 pt
allungamento interparola	1,66625 pt	spazio extra	1,11083 pt
accorciamento interparola	1,11083 pt	corpo nominale	10,00000 pt

Tabella 18.10: Il Text Companion Font con codifica TS1

18.6 Il Text Companion Font

LaTeX consente di usare un numero enorme di simboli, specialmente in matematica; ma da solo LaTeX non offre i simboli che si usano nel testo, dai segni delle unità monetarie ad altri segni che possono trovare posto in scritti di vario genere.

Così quando furono prodotti i primi font latini con la codifica T1, fu sentita immediatamente la necessità di un certo numero di segni o comandi non coperti con i simboli di nessun altro font, e fu predisposto il Text Companion Font, i cui simboli sono riportati nella tabella 18.10; ritorna per l'ennesima volta il segno

del mho, ℧, il cui unico pregio è quello di poter essere scritto in modo testo con il comando `\textmho`. Ahimè, trattandosi di un segno bandito dalle norme ISO, questo segno appare per la terza volta, ma inutilmente, perché non può essere usato.[7]

Il Text Companion Font veniva caricato e reso utilizzabile invocando il pacchetto *textcomp*. Le recenti modifiche (vedi il capitolo 30) al nucleo di LaTeX rendono superfluo caricare questo pacchetto, perché tutti i comandi, definiti da *textcomp*, vi sono migrati dentro. Anche se si usa il pacchetto *mathdesign* è superfluo caricare il pacchetto *textcomp*, perché tutti i simboli disponibili con i font Utopia, Charter e Garamond sono già tutti definiti.

18.7 Gli alfabeti diversi da quello latino

Solo pdflatex

Per scrivere in altre lingue con pdflatex bisogna avere invocato il pacchetto `babel` avendo indicato fra le opzioni le lingue che si vogliono usare; l'ultima lingua indicata nell'elenco delle opzioni diventa la lingua di default per l'intero documento. Dalla versione 3.9 di *babel*, la lingua principale può venire marcata con l'attributo *main* e in questo caso può apparire nella lista delle lingue in una posizione qualsiasi.

Quando si invoca una qualsiasi lingua che si scriva con l'alfabeto cirillico (russo, bulgaro, ucraino, eccetera) il pacchetto `babel` imposta direttamente anche l'uso dell'alfabeto cirillico. Una versione a 128 caratteri è riportata nella tabella 18.4, però con alcune di quelle lingue può venire scelta un'altra codifica, magari a 256 caratteri.

18.7.1 Caratteri cirillici

Solo pdflatex

Come già detto i font cirillici sono solitamente preinstallati in ogni distribuzione non minimale del sistema TeX. Le codifiche cirilliche, oltre alla già citata *OT2* a 128 caratteri, sono tre diverse versioni a 256 caratteri: *T2A*, *T2B* e *T2C*, e infine la codifica *X2*.

In che cosa differiscono queste codifiche? Nel fatto che la codifica a 128 caratteri richiede espressamente di essere attivata quando si usa il cirillico e permette di scrivere *esclusivamente* in russo. Le codifiche *T2x* hanno i primi 128 caratteri coincidenti con quelli della codifica *T1*, quindi sono caratteri latini, mentre i secondi 128 caratteri sono quelli cirillici validi per scrivere in russo, anche con ortografie 'prerivoluzionarie'. Ma la variante *A* serve per scrivere oltre che in caratteri latini anche con il cirillico usato nei paesi dell'Europa orientale (Ucraina, Bulgaria, Russia e altre). La variante *B* serve per comporre con i segni cirillici usati nelle repubbliche della Siberia Orientale. La variante *C* serve per scrivere in abcazio.

[7] In realtà è ancora usato negli Stati Uniti dove le norme ISO hanno meno seguito che in Europa. In Europa, invece, non dovrebbe più esserci nessuno che lo usi.

	’00	’01	’02	’03	’04	’05	’06	’07	’10	’11	’12	’13	’14	’15	’16	’17	
’000	̀ 0	́ 1	̂ 2	̃ 3	̈ 4	̋ 5	̊ 6	̌ 7	̆ 8	̄ 9	̇ 10	̧ 11	̨ 12	I 13	⟨ 14	⟩ 15	"00
’020	“ 16	” 17	̂ 18	̃ 19	̈ 20	– 21	— 22	23	̦ 24	̀ 25	́ 26	̄ 27	δ 28	δ 29	Ӈ 30	ӈ 31	"10
’040	⌣ 32	! 33	" 34	# 35	$ 36	% 37	& 38	’ 39	(40	) 41	* 42	+ 43	, 44	- 45	· 46	/ 47	"20
’060	0 48	1 49	2 50	3 51	4 52	5 53	6 54	7 55	8 56	9 57	: 58	; 59	< 60	= 61	> 62	? 63	"30
’100	@ 64	Æ 65	Ъ 66	Ħ 67	Ҽ 68	Ҿ 69	К 70	Қ 71	Ӆ 72	I 73	J 74	Љ 75	М 76	Њ 77	Ԛ 78	Ԓ 79	"40
’120	Р 80	Q 81	Т 82	S 83	Ҭ 84	Ц 85	Ч 86	W 87	Ѣ 88	Ӂ 89	V 90	[91	\ 92	] 93	^ 94	_ 95	"50
’140	‘ 96	æ 97	ђ 98	ħ 99	ҽ 100	ҿ 101	к 102	қ 103	ӆ 104	i 105	j 106	љ 107	м 108	њ 109	ԛ 110	ԓ 111	"60
’160	р 112	q 113	т 114	s 115	ҭ 116	ц 117	ч 118	w 119	ѣ 120	ӂ 121	v 122	{ 123		124	} 125	~ 126	- 127	"70
’200	Ґ 128	Ғ 129	Ӷ 130	Ҕ 131	Һ 132	Ӝ 133	Ӟ 134	Ӡ 135	Ї 136	Ҟ 137	Ҡ 138	Ҍ 139	Ӆ 140	Ӊ 141	Ҥ 142	Ӈ 143	"80
’220	Ө 144	Ҫ 145	Ў 146	Ү 147	Ұ 148	Х 149	Ҳ 150	Ҷ 151	Ҷ 152	Є 153	Ә 154	Ɛ 155	Ё 156	№ 157	¤ 158	§ 159	"90
’240	ґ 160	ғ 161	ӷ 162	ҕ 163	h 164	ж 165	ӟ 166	ӡ 167	ї 168	ҟ 169	ҡ 170	ҍ 171	ӆ 172	ӊ 173	ҥ 174	ӈ 175	"A0
’260	ө 176	ҫ 177	ў 178	ү 179	ұ 180	х 181	ҳ 182	ҷ 183	ч 184	є 185	ә 186	ɛ 187	ё 188	„ 189	« 190	» 191	"B0
’300	А 192	Б 193	В 194	Г 195	Д 196	Е 197	Ж 198	З 199	И 200	Й 201	К 202	Л 203	М 204	Н 205	О 206	П 207	"C0
’320	Р 208	С 209	Т 210	У 211	Ф 212	Х 213	Ц 214	Ч 215	Ш 216	Щ 217	Ъ 218	Ы 219	Ь 220	Э 221	Ю 222	Я 223	"D0
’340	а 224	б 225	в 226	г 227	д 228	е 229	ж 230	з 231	и 232	й 233	к 234	л 235	м 236	н 237	о 238	п 239	"E0
’360	р 240	с 241	т 242	у 243	ф 244	х 245	ц 246	ч 247	ш 248	щ 249	ъ 250	ы 251	ь 252	э 253	ю 254	я 255	"F0
	"00	"01	"02	"03	"04	"05	"06	"07	"08	"09	"0A	"0B	"0C	"0D	"0E	"0F	

Font parameters

inclinazione	0.0pt	x-height	4.3045pt
spazio interparola	3.33252pt	larghezza del quadrato	9.99756pt
allungamento interparola	1.66626pt	spazio extra	1.11084pt
accorciamento interparola	1.11084pt	corpo nominale	10.00000 pt

Tabella 18.11: Il font cirillico esteso con la codifica *X2*

È chiaro che queste tre codifiche permettono di scrivere contemporaneamente nelle lingue dell’Europa occidentale e in una o l’altra delle lingue dell’Europa orientale e dell’ex Unione Sovietica. Non tutte le collezioni di font, però, contengono sia il latino sia il cirillico dell’una o dell’altra specie e/o sono disponibili sia in forma vettoriale, come sarebbe desiderabile, sia in forma a matrice di punti.

Per scrivere testi cirillici senza i vincoli delle polizze e delle codifiche appena viste si può usare la codifica *X2*, la cui matrice di segni è mostrata nella tabella 18.11. Si noti che la distribuzione normale del sistema TEX, almeno fino alla distribuzione di TEX Live 2009, conteneva questi font adatti alla codifica *X2* solo a matrici di punti, non in formato scalabile. Se si fosse dovuto avere bisogno di questi font, non sarebbe stato un problema trasformarli in formato scalabile usando la procedura esemplificata nel paragrafo 18.8.4; con la distribuzione TEX Live 2010 non è più necessario ricorrere a questa trasformazione.

Va notata infine una cosa: i font cirillici con codifica *OT2* sono disponibili sia a matrici di punti sia in formato scalabile; questi font sono adatti a scrivere parole russe semplicemente cambiando codifica e impedendo la sillabazione; una tastiera latina è perfettamente in grado di immettere il testo russo ricorrendo alle sole lettere latine, un po’ come è stato descritto nel paragrafo 15.2 per scrivere in greco: in altre parole la disposizione dei segni cirillici nella tabella 18.4 e un set di legature permette di usare i segni latini e la traslitterazione latina del russo secondo la norma anglosassone, cosicché non è affatto difficile inserire qualche

parola russa all'interno del testo scritto in caratteri latini. Per esempio, dopo aver definito un comando `\textcyrillic` della forma[8]:

```
\newcommand\ciryllictext{\hyphenrules{nohyphenation}%
\fontencoding{OT2}\fontfamily{cmr}\selectfont}
\DeclareTextFontCommand{\textcyrillic}{\cyrillictext}
```

si può scrivere `\textcyrillic{Khrushchev}` dentro un testo scritto in lettere latine per ottenere Хрущев. Con le altre codifiche si possono fare cose analoghe; in particolare è possibile usare la codifica di entrata *utf8* e, disponendo di un editor adeguato per l'input dei caratteri cirillici, si può scrivere direttamente in cirillico con la certezza che quel testo viene composto in caratteri cirillici e il testo 'latino' in caratteri latini. Si può approfondire l'argomento nel bell'articolo di Enrico Gregorio che espone tutte le procedure nei minimi dettagli e mostra anche le soluzioni a qualche piccolo problema che si può manifestare, [28].

Per scrivere in russo, bulgaro, ucraino, eccetera, è meglio specificare la lingua come opzione a *babel*; così si dispone direttamente della sillabazione specifica della lingua, delle parole codificate per indicare l'equivalente di 'Capitolo', 'Figura', 'Tabella', eccetera, e per disporre dei comandi specifici per introdurre certi segni cirillici nel testo. Per la lingua russa, sono utilizzabili tutte le codifiche disponibili con il sistema TeX, ma quella suggerita è costituita dalla codifica *T2A*, sebbene quella di default, sembra di capire, sia la codifica *X2*. In ogni caso ognuna ha i propri vantaggi; la codifica suggerita è particolarmente utile per gli utenti russi che operano con tastiere russe, nelle quali con un semplice click si passa dai caratteri latini a quelli cirillici (e viceversa). Cosicché l'eventuale cambio di lingua avviene al livello della tastiera. Infatti sopra si è spiegato che i primi 128 caratteri della codifica *T2A* coincidono con i caratteri latini secondo la codifica ASCII (a 7 bit), mentre i secondi 128 caratteri contengono i caratteri cirillici validi per il russo; quindi comporre in modo misto non coinvolge il cambio del font di uscita. Ovviamente i caratteri ASCII non sono raccomandabili altro che per scrivere in inglese, forse l'unica lingua moderna occidentale priva di accenti.

lualatex e xelatex

Usando i programmi xelatex o lualatex basta impostare una qualunque lingua slava o asiatica scritta in cirillico come lingua principale o secondaria e definire una famiglia di font col nome *cyrillicfont* per scegliere un font OpenType che contenga il cirillico, e qualunque lingua da scriversi con quell'alfabeto viene composta con il font selezionato per il cirillico; gli ambienti *russian*, *bulgarian*, *ukranian*, eccetera sono già forniti da polyglossia in corrispondenza con le lingue specificate. Tutto ciò è molto più semplice rispetto a quando si usa pdflatex.

[8] Nel testo [28] lo stesso comando è definito in maniera molto più raffinata, nel senso che prende in considerazione la possibilità che la lingua *nohyphenation* non sia disponibile; qui il comando è definito supponendo di disporre di una distribuzione sufficientemente moderna del sistema TeX.

18.7.2 Caratteri greci

Quando usando pdflatex si invoca invece la lingua greca senza ulteriori specificazioni, viene anche impostato l'alfabeto greco riportato nella tabella 18.12 ma predisposto per la scrittura monotonica moderna, cioè con gli accenti ridotti al solo accento acuto nelle parole polisillabiche e dove sono eliminati gli spiriti; con queste semplificazioni diventa più frequente l'uso della dieresi per indicare gruppi di vocali che non formano dittonghi. Se si specifica invece l'attributo/modificatore `polutoniko` o l'attributo/modificatore `ancient` per la lingua greca diventano disponibili tutti i segni indicati nella tabella 18.12. Per usare il greco bisogna leggere attentamente la documentazione `babel` relativa a questa lingua per usare correttamente la transcodifica della tastiera USA (o anche italiana) nei caratteri greci; si noti che per la moltitudine di accenti e altri segni diacritici presenti con molte vocali, sono previste numerose legature; così la sequenza `>~a|` produce il segno ᾷ. Solo pdflatex

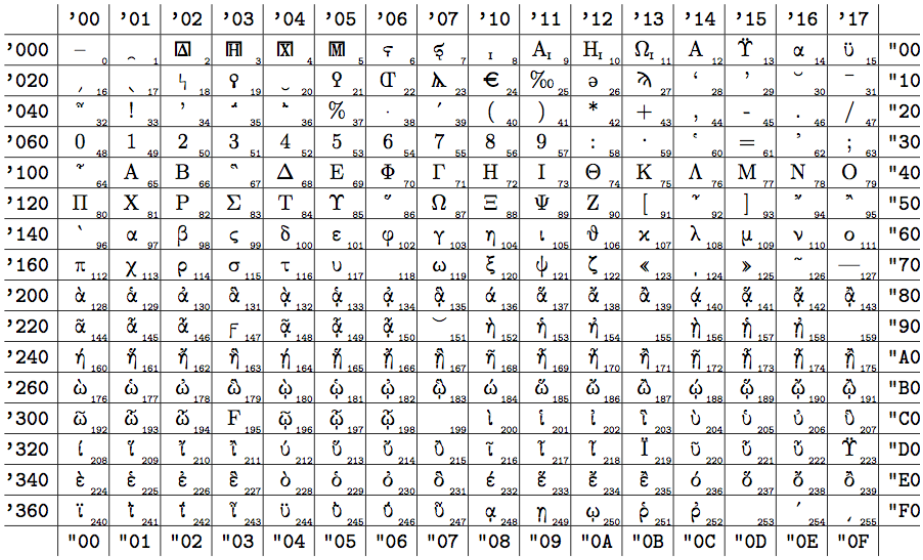
Pur essendo disponibili tutti i comandi di *babel* per comporre brani di testo in una lingua differente da quella principale, per il greco esistono diversi pacchetti che consentono di fare cose simili anche senza avere invocato la lingua greca. Un pacchettino molto utile (che però richiede la specificazione della lingua greca,) è *begingreek* che definisce l'ambiente *greek* per scrivere un brano di testo più o meno lungo in greco, oppure il comando `\greektxt` per scrivere una parola o una breve locuzione; entrambi l'ambiente e il comando accettano un argomento facoltativo per specificare la famiglia di font greci che si preferisce usare; per esempio si può scrivere nel file sorgente

```
\greektxt[artemisia]{t\'eqnh}
```

per ottenere τέχνη, quando senza l'argomento facoltativo si otterrebbe τέχνη. Per ulteriori informazioni circa l'uso normale dei font greci si veda il paragrafo 15.2.

Per produrre testi greci finemente commentati con le notazioni filologiche del settore si può estendere la funzionalità di LaTeX con il pacchetto *teubner* che consente di introdurre una grande varietà di segni specifici, di usare una serie di font greci inclinati particolarmente leggibili, sviluppati preso la tipografia Teubner di Lipsia e in onore della quale è stato scelto il nome del pacchetto. Tuttavia va fatto osservare che pacchetti di vario genere possono collidere; in particolare il pacchetto *teubner* inizialmente collideva con i pacchetti *amsmath* e compagni; questo comportamento è stato corretto nel febbraio 2008 grazie alle indicazioni di un utente. È vero che difficilmente in un lavoro di filologia classica greca c'è bisogno di usare matematica avanzata, tuttavia questo problema esiste e in questi casi l'utente che si scontra con problemi derivanti dal conflitto di pacchetti dovrebbe avere 'il senso civico' di avvisare i curatori dei pacchetti, come avviene regolarmente con tutto il software libero, il cui progresso avviene proprio grazie a questa interazione fra programmatori e utenti.

Recentemente il pacchetto *teubner* è stato arricchito di altre funzionalità che permettono di comporre i testi filologici in modo tipograficamente migliore. In particolare diversi segni, che era stato necessario creare con soluzioni non



Parametri			
inclinazione	0,00000	x-height	4,30450 pt
spazio interparola	3,33251 pt	larghezza del quadrato	9,99755 pt
allungamento interparola	1,66625 pt	spazio extra	1,11083 pt
accorciamento interparola	1,11083 pt	corpo nominale	10,00000 pt

Tabella 18.12: Il font greco a 256 caratteri con codifica LGR

molto azzeccate, sono stati corretti sfruttando le proprietà grafiche del motore di composizione pdflatex; allo stesso tempo le crenature dei segni accentati che con le macro precedenti richiedevano non poca ingegnosità per essere applicate correttamente, ora possono essere affrontate con macro che sfruttano meglio i comandi interni di pdflatex. Speriamo dunque che i filologi si possano trovare meglio con la nuova versione.

Il pacchetto *teubner* per il momento non funziona ancora per lavorare con xelatex e i font OpentType.

Solo xelatex

Usando xelatex non ci sono differenze rispetto all'uso di altre lingue; basta, dopo aver caricato *polyglossia* specificare la lingua greca come lingua principale o come lingua secondaria, ricordando di specificare la variante fra *mono*, *poly* o *ancient*, tenendo conto che la mancanza della variante implica l'uso dell'ortografia e della sillabazione monotonica. Conviene anche definire dopo avere caricato *fontspec* una famiglia di font OpenType che contenga il greco; specificandone il nome mediante l'istruzione `greekfont`, cosicché quando si vuole scrivere in greco, in particolare con l'ambiente greek definito da *polyglossia*, viene automaticamente selezionato quel font.

Vale la pena segnalare che la lingua greca può essere specificata con varianti

diverse lungo lo sviluppo del documento da comporre; questa probabilmente è una funzionalità interessante, per esempio, per chi scrive in greco moderno per riportare brani di greco antico.

Solo lualatex

Usando lualatex si procede come quando si usa xelatex con una sola sostanziale differenza; infatti lualatex, per impostazione predefinita, carica per ora un solo file di pattern di sillabazione per una data lingua, perché questo programma carica questi file in tempo reale sulla base delle lingue usate in un dato documento; invece xelatex richiede che tutti i pattern di sillabazione siano precaricati nel file di formato. Ci sono vantaggi e svantaggi nei due approcci, ma in questo caso lualatex è svantaggiato per lavorare con il greco e le sue tre varianti che implicano tre differenti file di pattern di sillabazione. Per questo lualatex carica soltanto i pattern per il greco monotonico; nessuno impedisce di scrivere il testo sorgente anche o solo in greco politonico o in greco antico, ma la sillabazione sarà inevitabilmente carente o errata. Dal 2018 questo problema non dovrebbe essere stato eliminato.

Ovviamente non si può ancora usare il pacchetto *teubner* per l'impossibilità di questo di lavorare con i font OpenType e per il problema della sillabazione monotonica con vecchie versioni del sistema TEX. Sembra che ora si stia lavorando per estendere il pacchetto *teubner* per renderlo usabile anche con questi programmi di composizione.

18.7.3 Scrivere con altri alfabeti

È interessante vedere come il sistema TEX disponga di una miriade di caratteri anche per lingue insolite e consenta di comporre in lingue che fanno uso di questi alfabeti.

Nella figura 18.7 vi è rappresentato il Credo cristiano come stabilito nel Concilio di Nicea; esso è composto in latino, in greco e in copto nella varietà bohairica, quella che viene usata nei testi liturgici della chiesa copta. Il latino accentato è la normale versione del latino ecclesiastico per il quale esiste il pacchetto *ecclesiastic* (o il modificatore/variante *ecclesiastic* per il modulo *latin* per *babel* e *polyglossia*). Per il greco si è già detto. Per il copto esiste il pacchetto *coptic* che però consente di comporre quella lingua nella varietà sahidica; il pacchetto *bohairic* esiste ma oggi (2017) non è ancora disponibile per la generalità degli utenti. Le due versioni differiscono solo per la presenza delle lettere accentate, che per la liturgia copta svolgono più o meno la stessa funzione delle lettere accentate usate per il latino ecclesiastico.

Non si ritiene opportuno di riportare il codice sorgente, perché sarebbe molto complicato e di scarsissimo interesse per la maggior parte dei lettori. Inoltre si potrebbero portare esempi anche con scritti in arabo o in ebraico; si potrebbero portare esempi con brevi testi in cinese o giapponese o coreano; in lingue del tutto esotiche come la lingua inuhit degli eschimesi, e tanto altro. Ogni lingua dispone o può disporre di font adatti alla sua composizione, e questa è una delle forze del sistema TEX e dei suoi vari motori di composizione.

Il Credo di Nicea in tre lingue

Credo in unum Deum, Patrem omnipoténtem Factórem cæli et terræ, visibílium ómnium et invisibílium.

Et in unum Dóminum Iesum Christum, Fílium Dei Unigénitum, et ex Patre natum ante ómnia sæcula. Deum de Deo, lumen de lúmine, Deum verum de Deo vero, génitum, non factum, consubstantiálem Patri: per quem ómnia facta sunt.

Qui propter nos hómines et propter nostram salútem descéndit de cælis. Et incarnátus est de Spíritu Sancto ex María Vírgine, et homo factus est.

Crucifíxus étiam pro nobis sub Póntio Piláto; passus, et sepúltus est, et resurréxit tértia die, secúndum Scriptúras, et ascéndit in cælum, sedet ad déxteram Patris. Et íterum ventúrus est cum glória, iudicáre vivos et mórtuos, cuius regni non erit finis.

Et in Spíritum Sanctum, Dóminum et vivificántem: qui ex Patre (Filióque) procédit. Qui cum Patre et Fílio simul adorátur et conglorificátur: qui locútus est per prophétas. Et unam, sanctam, cathólicam et apostólicam Ecclésiam. Confíteor unum baptísma in remissiónem peccatorum. Et expecto resurrectionem mortuorum, et vitam ventúri sæculi. Amen.

Πιστεύομεν εἰς ἕνα Θεόν, Πατέρα, Παντοκράτορα, ποιητὴν οὐρανοῦ καὶ γῆς, ὁρατῶν τε πάντων καὶ ἀοράτων.

Καὶ εἰς ἕνα Κύριον Ἰησοῦν Χριστόν, τὸν Υἱὸν τοῦ Θεοῦ τὸν μονογενῆ, τὸν ἐκ τοῦ Πατρὸς γεννηθέντα πρὸ πάντων τῶν αἰώνων· φῶς ἐκ φωτός, Θεὸν ἀληθινὸν ἐκ Θεοῦ ἀληθινοῦ, γεννηθέντα οὐ ποιηθέντα, ὁμοούσιον τῷ Πατρί, δι᾽ οὗ τὰ πάντα ἐγένετο.

Τὸν δι᾽ ἡμᾶς τοὺς ἀνθρώπους καὶ διὰ τὴν ἡμετέραν σωτηρίαν κατελθόντα ἐκ τῶν οὐρανῶν καὶ σαρκωθέντα ἐκ Πνεύματος Ἁγίου καὶ Μαρίας τῆς Παρθένου καὶ ἐνανθρωπήσαντα.

Σταυρωθέντα τε ὑπὲρ ἡμῶν ἐπὶ Ποντίου Πιλάτου, καὶ παθόντα καὶ ταφέντα. Καὶ ἀναστάντα τῇ τρίτῃ ἡμέρα κατὰ τὰς Γραφάς. Καὶ ἀνελθόντα εἰς τοὺς οὐρανοὺς καὶ καθεζόμενον ἐκ δεξιῶν τοῦ Πατρός. Καὶ πάλιν ἐρχόμενον μετὰ δόξης κρῖναι ζῶντας καὶ νεκρούς, οὗ τῆς βασιλείας οὐκ ἔσται τέλος.

Καὶ εἰς τὸ Πνεῦμα τὸ Ἅγιον, τὸ κύριον, τὸ ζωοποιόν, τὸ ἐκ τοῦ Πατρὸς ἐκπορευόμενον, τὸ σὺν Πατρὶ καὶ Υἱῷ συμπροσκυνούμενον καὶ συνδοξαζόμενον, τὸ λαλῆσαν διὰ τῶν προφητῶν. Εἰς μίαν, Ἁγίαν, Καθολικὴν καὶ Ἀποστολικὴν Ἐκκλησίαν. Ὁμολογῶ ἓν βάπτισμα εἰς ἄφεσιν ἁμαρτιῶν. Προσδοκῶ ἀνάστασιν νεκρῶν. Καὶ ζωὴν τοῦ μέλλοντος αἰῶνος. Ἀμήν.

Ⲡⲉⲛⲛⲁϩ ⲉⲟⲩⲛⲟⲩϯ ⲛⲟⲩⲱⲧ: Ⲫⲛⲟⲩϯ Ⲫⲓⲱⲧ ⲡⲓⲡⲁⲛⲧⲟⲕⲣⲁⲧⲱⲣ: ⲫⲏⲧⲁϥϣⲁⲓⲇ ⲛ̀ⲧⲉ ⲛⲉⲙ ⲛ̀ⲕⲁϩⲓ: ⲛⲏⲧⲟⲩⲛⲁⲩ ⲉ̀ⲣⲱⲟⲩ ⲛⲉⲙ ⲛⲏⲧⲉ ⲛ̀ⲥⲉⲛⲁⲩ ⲉ̀ⲣⲱⲟⲩ ⲁⲛ.

Ⲡⲉⲛⲛⲁϩ ⲉⲟⲩⲃⲟⲓⲥ ⲛⲟⲩⲱⲧ Ⲓⲏⲥⲟⲩⲥ Ⲡⲓⲭ̀ⲣⲓⲥⲧⲟⲥ Ⲡ̀ϣⲏⲣⲓ ⲙ̀Ⲫⲛⲟⲩϯ ⲡⲓⲙⲟⲛⲟⲅⲉⲛⲏⲥ: ⲡⲓⲙⲓⲥⲓ ⲉ̀ⲃⲟⲗϩⲉⲛ Ⲫⲓⲱⲧ ϩⲁϫⲱⲟⲩ ⲛ̀ⲛⲓⲉⲱⲛ ⲧⲏⲣⲟⲩ: ⲟⲩⲟⲩⲱⲓⲛⲓ ⲉ̀ⲃⲟⲗϩⲉⲛ ⲟⲩⲟⲩⲱⲓⲛⲓ: ⲟⲩⲛⲟⲩϯ ⲛ̀ⲧⲁϥ̀ⲱⲙⲓ ⲉ̀ⲃⲟⲗϩⲉⲛ ⲟⲩⲛⲟⲩϯ ⲛ̀ⲧⲁϥ̀ⲱⲙⲓ: ⲟⲩⲙⲓⲥⲓ ⲡⲉ ⲟⲩⲑⲁⲙⲓⲇ ⲁⲛ ⲡⲉ: ⲟⲩⲙⲟⲟⲩⲥⲓⲟⲥ ⲡⲉ ⲛⲉⲙ Ⲫⲓⲱⲧ: ⲫⲏⲧⲁ ϩⲱⲃ ⲛⲓⲃⲉⲛ ⲱⲟⲡⲓ ⲉ̀ⲃⲟⲗϩⲓⲧⲟⲧϥ.

Ⲫⲁⲓ ⲉ̀ⲧⲉ ⲉⲃⲏⲏⲧⲉⲛ ⲛ̀ⲛⲟⲛ ⲇⲁ ⲛⲓⲣⲱⲙⲓ ⲛⲉⲙ ⲉⲃⲏ ⲡⲉⲛⲟⲩϫⲁⲓ: ⲁϥⲓ̀ ⲉ̀ⲡⲉⲥⲏⲧ ⲉ̀ⲃⲟⲗϩⲉⲛ ⲧ̀ϥⲉ: ⲁϥⲃⲓⲥⲁⲣϫ ⲉ̀ⲃⲟⲗϩⲉⲛ Ⲡⲓⲡ̀ⲛⲉⲩⲙⲁ ⲉⲃⲟⲩⲁⲃ ⲛⲉⲙ ⲉ̀ⲃⲟⲗϩⲉⲛ Ⲩⲁⲣⲓⲁ ⲧ̀ⲡⲁⲣⲑⲉⲛⲟⲥ ⲟⲩⲟⲩ ⲁϥⲉ̀ⲣⲣⲱⲙⲓ.

Ⲟⲩⲟⲩ ⲁⲧⲉⲣⲉ̀ⲧⲁⲧⲣⲱⲛⲓⲛ ⲙ̀ⲙⲟⲩ ⲉ̀ⲃⲏⲣⲓ ⲉ̀ϫⲱⲛ ⲛⲁⲩⲣⲉⲛ Ⲡⲟⲛⲧⲓⲟⲥ Ⲡⲓⲗⲁⲧⲟⲥ: ⲁϥⲱⲉⲡⲉⲙⲕⲁϩ ⲟⲩⲟⲩ ⲁⲧⲕⲟⲥⲩ. Ⲟⲩⲟⲩ ⲁϥⲧⲱⲛϥ ⲉ̀ⲃⲟⲗϩⲉⲛ ⲛⲏⲉⲑⲙⲱⲟⲩⲧ ϩⲉⲛ ⲡⲓⲉ̀ⲅⲟⲟⲩ ⲙ̀ⲙⲁⲩⲱⲟⲩⲧ ⲕⲁⲧⲁ ⲛⲓⲃ̀ⲣⲁⲫⲏ. Ⲇ̀ⲭⲱⲉⲛⲁϥ ⲉ̀ⲛ̀ϣⲱⲓ ⲉ̀ⲛⲓⲫⲏⲟⲩⲓ̀: ⲁϥⲉⲙⲉⲓ ⲥⲁⲟⲩⲓ̀ⲛⲁⲓ ⲙ̀ⲡⲉϥⲱⲧ. Ⲕⲉ ⲡⲁⲗⲓⲛ ϣⲛⲏⲟⲩ ϩⲉⲛ ⲡⲉϥⲱⲟⲩ ⲉ̀ϯϩⲁⲡ ⲉⲛⲛⲉⲧⲟⲛϩ ⲛⲉⲙ ⲛⲏⲉⲑⲙⲱⲟⲩⲧ: ⲫⲏⲧⲉ ⲧⲉϥⲙⲉⲧⲟⲩⲣⲟ ⲟⲩⲁⲑⲙⲟⲩⲛⲕ ⲧⲉ.

Ⲥⲉ ⲧⲉⲛⲛⲁϩ ⲉⲡⲓⲡ̀ⲛⲉⲩⲙⲁ ⲉⲃⲟⲩⲁⲃ: Ⲛ̀ⲃⲟⲓⲥ ⲛ̀ⲣⲉϥϯ ⲙ̀ⲡⲱⲛϩ: ⲫⲏⲉⲑⲛⲏⲟⲩ ⲉ̀ⲃⲟⲗϩⲉⲛ Ⲫⲓⲱⲧ: ⲥⲉⲟⲩⲱϣⲧ ⲙ̀ⲙⲟⲩ ⲥⲉϯⲱⲟⲩ ⲛⲁϥ ⲛⲉⲙ Ⲫⲓⲱⲧ ⲛⲉⲙ Ⲡ̀ϣⲏⲣⲓ: ⲫⲏⲧⲁϥⲥⲁⲭⲓ ϩⲉⲛ ⲛⲓⲡ̀ⲣⲟⲫⲏⲧⲏⲥ. Ⲉⲟⲩⲏ ⲛ̀ⲁⲗⲓⲁ ⲛ̀ⲕⲁⲑⲟⲗⲓⲕⲏ ⲛ̀ⲁⲡⲟⲥⲧⲟⲗⲓⲕⲏ ⲛⲉⲕⲕⲗⲏⲥⲓⲁ. Ⲡⲉⲛⲉⲣⲑⲟⲙⲟⲗⲟⲅⲓⲛ ⲛⲟⲩⲱⲙⲥ ⲛⲟⲩⲱⲧ ⲉ̀ⲡ̀ϫⲱ ⲉ̀ⲃⲟⲗ ⲛ̀ⲧⲉ ⲛⲓⲛⲟⲃⲓ. Ⲡⲉⲛϫⲟⲩⲱϯ ⲉ̀ⲃⲟⲗ ⲇⲁⲛ̀ϩ ⲛ̀ⲧⲁⲛⲁⲥⲧⲁⲥⲓⲥ ⲛ̀ⲧⲉ ⲛⲓⲣⲉϥⲙⲱⲟⲩⲧ: ⲛⲉⲙ ⲡⲓⲱⲛϩ ⲛ̀ⲧⲉ ⲡⲓⲉ̀ⲱⲛ ⲉⲟⲛⲛⲟⲩ: ⲁⲙⲏⲛ.

Figura 18.7: Il Credo nisseno-costantinopolitano in tre lingue (latino, greco, copto) che fanno uso di alfabeti diversi.

Si tenga presente che il sistema TEX con i suoi diversi motori di composizione permette di comporre testi in un centinaio di lingue. Per ciascuna di solito dispone di font adeguati e di moduli per `babel` o `polyglossia` che provvedono ai font necessari per comporre in ciascuna di quelle lingue. Ovviamente ciò non impedisce agli utenti di scegliere font diversi da quelli preimpostati; questa operazione di solito non è facilissima con pdflatex ma è quasi banale con xelatex e lualatex.

18.8 La gestione dei font

Gestire la moltitudine di font sul proprio elaboratore può talvolta risultare faticoso; bisogna tenere presente che, tradizionalmente, i font classici di LATEX sono per lo più distribuiti sotto forma di sorgenti scritti con il linguaggio di programmazione METAFONT, e che tutti i programmi delle distribuzioni del sistema TEX sono in grado di lanciare METAFONT, il programma, in modo da generare al volo i font in formato bitmapped adatti al tipo di schermo e di stampante che si sono configurati durante l'installazione del sistema.

Ma questi font bitmapped sono buoni per la stampa, ma non lo sono per la visualizzazione a schermo, come si è già fatto notare. Inoltre, benché non sia impossibile usarli con xelatex e lualatex essi erano pensati inizialmente per tex, poi per latex; ma oggi con pdflatex, xelatex e lualatex si preferiscono di gran lunga i font vettoriali. Lo si è già detto, ma bisogna ripeterlo: pdflatex gestisce solamente i font a 256 caratteri (codificati con indirizzi a 8 bit), mentre xelatex e lualatex gestiscono anche e preferibilmente i font OpenType e TrueType codificati in UNICODE, con indirizzi di diversi byte.

Nel seguito si cercherà di tenere distinte le due classi di font, codificati con un byte, o con molti byte, perché la loro gestione è diversa. Si tengano perciò sott'occhio le note marginali con le quali si marcano quanto è valido solo per i font codificati con un byte validi per pdflatex e invece ciò che è valido per i font gestibili solo con xelatex e lualatex.

Ci si ricordi che mentre su carta i font bitmapped creati per la densità di stampa della stampante in uso danno luogo ad un risultato perfetto, quegli stessi font non sono il meglio che si possa usare per leggere il testo sullo schermo, specialmente se si desidera leggere il testo con ingrandimenti diversi.

Sarebbe meglio disporre dei font in formato vettoriale, ma questi non sono facili da ottenere dai file sorgente METAFONT, quindi è preferibile scaricarli dalla rete 'già fatti'. Inoltre c'è la miriade di font che si possono comperare o che si trovano come freeware, sia in formato Adobe Type 1 (PostScript), sia in formato TrueType e in formato OpenType. La loro installazione non è banale per i seguenti motivi.

LATEX per lavorare non ha bisogno dei disegni dei font, ma delle loro caratteristiche geometriche; queste sono raccolte in file con l'estensione `.tfm` (TEX Font Metrics); se si usa latex sarà poi il driver che trasforma il file DVI in forma leggibile quello che avrà bisogno dei disegni che, in formato bitmapped,

hanno estensioni del tipo .⟨*dpi*⟩pk, in formato PostScript .pfb (PostScript Font in Binary format), in formato TrueType .ttf (TrueType Font), in formato OpenType .otf (OpenType Font). Se invece si usano gli altri programmi di composizione, essi hanno bisogno simultaneamente sia dei file metrici sia dei file con le descrizioni vettoriali dei contorni o, in mancanza, dei file bitmapped.

I font bitmapped di origine METAFONT sono codificati solo con un byte e vengono generati insieme con il file .tfm, quindi non c'è bisogno d'altro.

I font PostScript ottenuti dal mercato o dalla rete (non ottenuti trasformando i font bitmapped, che ne sono già dotati) sono disponibili con il file metrico che ha formato diverso dal .tfm ed ha estensione .afm (Adobe Font Metrics).

I font TrueType e i font OpenType non dispongono di un file metrico separato, perché le informazioni metriche sono già contenute nel file stesso che contiene i disegni dei font.

In questa situazione bisogna leggere attentamente le guide che accompagnano il sistema TeX della propria distribuzione e seguirle scrupolosamente; i programmi per le debite conversioni di formato dei file metrici fanno già parte della distribuzione, ma le operazioni da fare devono esser svolte dall'utente/amministratore del PC con grande cura.

La raccomandazione di leggere la documentazione in questo caso è più che una raccomandazione: è un obbligo!

Però questa difficoltà è compensata dal fatto che il sistema TeX, contrariamente a quel che si mormora da parte di coloro che non leggono la documentazione, non è affatto vincolato ai suoi font METAFONT, ma può lavorare con qualunque font esista al mondo, certamente, se non tutti, ameno un numero sterminato.

Infine va ricordato che gli ultimi nati della famiglia dei programmi di composizione del sistema TeX, xelatex e lualatex, consentono di lavorare con i font con la codifica UNICODE e sanno trovare da soli le informazioni metriche di cui necessitano sia nei file .tfm o .afm sia all'interno dei file .ttf e .otf, quindi l'uso dei font e degli alfabeti più disparati (compresi quelli ad ideogrammi) diventa decisamente semplificata, specialmente se si dispone di uno *shell editor* che possa scrivere il testo sorgente facendo uso direttamente della codifica UNICODE. Anche ConTeXt-MK-IV (che oggi usa come motore di composizione luatex) è capace di lavorare con i font OpenType con codifica UNICODE.

Va fatta una precisazione: il sistema TeX viene distribuito dalla rete oppure mediante dischi di installazione, ma spesso l'utente installa solo la versione base arricchita dai pacchetti di estensione di cui necessita via via che usa questo sistema di composizione; sarebbe sempre meglio installare la versione completa, ma l'utente non installa tutto perché il sistema e l'intera collezione di estensioni è enorme e quindi piuttosto ingombrante sul disco. Sui PC e laptop più recenti questo non è un vero problema, perché 3 GiB o 4 GiB di occupazione di memoria sono una inezia nei dischi rigidi e nei dischi a stato solido di oggi.

L'installazione parziale è particolarmente infelice per quanto riguarda i pacchetti di estensione che arricchiscono le potenzialità compositive del sistema mediante una grande collezione di font; i diversi formati, però, fanno sì che

siano necessarie delle conversioni, perché non tutti i programmi usano lo stesso formato.

Inoltre, come si è avuto occasione di commentare più volte, le codifiche dei font possono essere diverse, spesso lo sono; ma TEX conosce solo le sue codifiche, quindi, talvolta è necessario anche provvedere al cambiamento di codifica. Per questo scopo si usano sia i 'font virtuali' sia i 'vettori di codifica'. Il discorso, come si vede, può diventare assai complicato, e non si procede oltre.

Tuttavia, per rendersi conto del perché certe cose non sono fatte come uno si aspetterebbe, bisogna ricordare che i file DVI possono essere mostrati su schermo da programmi come YAP o xdvi; e questi quando devono rappresentare graficamente i singoli segni dei font ricorrono di default alle loro immagini bitmapped.

Altri programmi facenti parte del sistema TEX, come per esempio dvips (che trasforma i file dal formato DVI al formato PostScript) oppure dvipdfmx (che trasforma i file dal formato DVI al formato PDF) preferiscono usare i font vettoriali e ricorrono a quelli bitmapped solo in caso di necessità. Hanno perciò bisogno di sapere quali font vettoriali siano disponibili e dove si trovino, perché questa informazione non è contenuta nel file DVI. Il programma pdflatex, invece, ha bisogno di conoscere subito quali font siano disponibili, in che formato e dove si trovino questi font, perché nel file sorgente non c'è nessun accenno a questa informazione. I programmi xelatex e lualatex sono più autonomi e non hanno bisogno di queste informazioni. Ma, attenzione, non dimentichiamo che quanto è disponibile su una macchina non è necessariamente disponibile su un'altra, perché il suo amministratore non ha caricato gli stessi pacchetti e/o non ha comperato gli stessi prodotti commerciali.

Solo pdflatex e latex

Ecco allora che sono necessari tre file per svolgere le funzioni appena descritte; essi si chiamano `psfonts.map`, `dvipdfm.map` e `pdftex.map`; sono collocati in tre cartelle diverse dell'albero di sistema o dell'albero personale delle cartelle di TEX; essi contengono le opportune informazioni rispettivamente per dvips, o per dvipdfmx, o per pdftex (l'interprete di pdflatex).

Durante la lavorazione di questo testo le persone che hanno contribuito si sono rese conto che la loro configurazione, benché fortemente personalizzata e piuttosto ricca di estensioni, tuttavia rendeva difficile il lavoro di gruppo, e qualche volta lo rendeva impossibile. Per questo inizialmente si è scelto di usare i font Computer Modern con codifica OT1, anche se non è la scelta ottimale per le lingue con gli accenti come l'italiano, ma è sicuramente presente su ogni installazione. In realtà da diversi anni per le successive versioni di questa Guida si usano i font Latin Modern con la codifica T1, ma il master file dice esattamente che cosa fare se questi font non fossero installati. Finora nessun collaboratore si è lamentato di questi font ed è un bene, perché significa che persino le distribuzioni base fortemente personalizzate sono in grado di essere uniformi con questi font vettoriali, forse persino un poco più belli dei font Computer Modern, ma certamente più adatti alla lingua italiana che fa uso relativamente spesso di diversi accenti.

18.8.1 Font OpenType

lualatex o xelatex

I font OpenType codificati con più byte, perché usano la codifica UNICODE, distribuiti con il sistema TEX si trovano nella cartella `.../texmf-dist/fonts/opentype/`. Alcune installazioni sono tali per cui il sistema operativo è immediatamente conscio della loro presenza che si aggiunge alla disponibilità di altri font OpenType presenti sul disco fisso grazie al sistema operativo stesso, o all'installazione di alcuni applicativi, come certe suite di programmi per ufficio.

Per altri sistemi operativi, come varie versioni di Linux, si rende nota la presenza dei font OpenType installati con il sistema TEX semplicemente rinfrescando la 'cache' dei font; come si faccia dipende dalla particolare versione di Linux, ma tipicamente bisogna dare una tantum (vale a dire ogni volta che si installa o si reinstalla il sistema TEX) un comando da linea di comando che informa il sistema operativo della necessita di aggiornare la 'cache' dei font, e il sistema operativo si mette al lavoro per eseguire questa operazione; essa dura qualche decina di secondi, ma deve essere eseguita, al massimo, solo una volta all'anno.

Per il sistema operativo Mac OS X delle macchine Apple, bisogna selezionare tutti i font contenuti nella citata cartella `.../opentype/` e aprirli con l'applicativo FontBook (o LibroFont), il quale chiederà via via conferma, ma installerà tutti i font della cartella, verificandone la validità e aggiungendoli non solo alla 'cache' dei font, ma anche rendendoli visibili nella cartella `~/Library/Fonts/`.

Ciò fatto tutti i font UNICODE già installati con il sistema operativo o con altri applicativi, sono immediatamente disponibili per l'uso con xelatex e lualatex. Si noterà però che ogni volta che si aggiorna un file OpenType o il sistema operativo, benché la 'cache' dei font sia già completa, il programma lualatex diventa un po' lento e non sembra lavorare bene; in realtà esso si aggiorna la sua personale 'cache', ma superata questa fase iniziale, si rimette a funzionare con la sua solita velocità di elaborazione, e non subisce più questi ritardi se non, appunto, quando avvengono quegli aggiornamenti.

Quali font sono allora disponibili su ciascuna delle nostre macchine? Come si fa ad averne un elenco? Chi scrive può dirlo solo per le macchine Apple, perché lavora con una di queste e non dispone di altre macchine né reali né virtuali. I dettagli sono quindi diversi da macchina a macchina, come lo sono i programmi da usare, ma sostanzialmente le azioni da eseguire sono le stesse su tutte le macchine.

Se quindi io apro la cartella `~/Library/Fonts/` sulla mia macchina e seleziono la visione di tutti i font OpenType (non solo quelli di sistema o quelli che mi sono installato personalmente compresi quelli del sistema TEX), posso vedere un elenco di circa 400 famiglie di font OpenType; ogni famiglia contiene generalmente quattro varietà, Regular, Bold, Italics, Bolditalics, ma alcune, poche famiglie ne contengono solo una, mentre altre ne possono contenere una dozzina. Va notato che i font con l'estensione `.otf` sono i 'normali' font OpenType, mentre quelli con l'estensione `.ttf` si riferiscono a font TrueType codificati UNICODE. Questo dipende dal fatto che i font TrueType hanno i contorni descritti da curve di

Bézier di secondo grado (archi di parabole) mentre i veri font OpenType hanno i contorni descritti da curve di Bézier di terzo grado (archi di curve cubiche); queste sono finezze che all'utente normale non interessano, quindi non ci si preoccupi se alcuni file di font hanno un'estensione e altri ne hanno un'altra.

Alcuni font hanno nomi arcinoti: Latin Modern, Computer Modern, CM-super, Times, Palatino, Arial, Verdana, Helvetica, Courier, Garamond, Baskerville, Bodoni, Utopia, Palladio, XITS e STIX, Libertine, Libertinus, e via di questo passo. Fra i font appartenenti alla distribuzione del sistema TEX, oltre ai primi tre nominati sopra, ci sono gli Antyqwa Torunska, i font della collezione TeX Gyre, che sono una rivisitazione di font commerciali ben noti riveduti e corretti dal gruppo dei Soci del GUST, gruppo degli utenti polacchi, i font messi a disposizione dalla Società dei Tipografi Ellenici GFSDidot, GFSBodoni, GFSArtemisia ed altri; insomma senza andare a cercare font rarissimi, l'utente del sistema TEX ha a disposizione centinaia di font, che gli consentono di scrivere in qualsiasi alfabeto quasi qualunque lingua parlata e, specialmente, scritta nel mondo. Come si è detto altrove, le lingue che può gestire il sistema TEX nelle sue varianti ammontano a un centinaio.

Tutti i font citati dispongono dei caratteri latini estesi, quasi tutti contengono anche il cirillico e il greco estesi; un buon numero contiene alcuni alfabeti retrogradi, scritti da destra a sinistra, ebraico, arabo, farsi, eccetera; altri contengono anche gli ideogrammi cinesi e i sillabari giapponesi e coreano; inoltre alfabeti meno conosciuti come il georgiano, il devanagari, il thai, eccetera.

Come si fa a usare tutti questi font? È molto semplice e si sono mostrati già diversi esempi; per riassumere: bisogna caricare il file di estensione *fontspec* specificandogli solo opzioni che valgano per tutto il documento; chi scrive usa sempre le legature TEX, quindi specifica sempre la chiamata nel modo seguente:

```
\usepackage{fontspec}[Ligatures=TeX]
```

Questo pacchetto mette a disposizione degli utenti i comandi per impostare le tre famiglie principali e il comando per caricare un font al quale si richiedono caratteristiche speciali:

```
\setmainfont{⟨famiglia con grazie⟩}[⟨opzioni⟩]
\setsansfont{⟨famiglia senza grazie⟩}[⟨opzioni⟩]
\setmonofont{⟨famiglia monospaziata⟩}[⟨opzioni⟩]

\fontspec{⟨font particolare⟩}[⟨opzioni⟩]
\newfontfamily{⟨comando⟩}{⟨font particolare⟩}[⟨opzioni⟩]

\usepackage{amsmath} % facoltativo ma consigliabile sempre
\usepackage{unicode-math}
\setmathfont{⟨font matematico⟩}[⟨opzioni⟩]
```

I nomi degli argomenti da specificare sono autoesplicativi, ma con l'ovvio commento che le ⟨*opzioni*⟩ specificate in ogni riga non sono dello stesso genere.

Invece ⟨*comando*⟩ è un comando formato dal solito backslash seguito da una stringa letterale. Se dopo avere caricato `fontspec` e `polyglossia` e dopo aver specificato i font normali per le tre famiglie principali e le lingue da usare, si definisce una nuova famiglia con il nome del ⟨*comando*⟩ ottenuta agglutinando il nome di una lingua con la parola font, \⟨*lingua*⟩`font`, allora questa famiglia verrà usata ogni volta che si scrive testo in quella ⟨*lingua*⟩. Questa è una funzionalità comodissima.

Si noti che nella sintassi appena mostrata viene anche indicato come gestire i font per la matematica, con l'annotazione che se si vuole usare il file di estensione `amsmath`, lo si può fare, ma lo si deve caricare prima del file di estensione `unicode-math` e della scelta del ⟨*font matematico*⟩. Si noti infine che i font OpenType possono contenere migliaia di glifi, per cui usando lualatex e xelatex non ci si deve preoccupare dei *math group*, o famiglie matematiche, né del loro numero limitato a 16, come avviene per pdflatex; ciascun font matematico di tipo OpenType contiene ogni alfabeto possibile, ogni serie e ogni forma possibile e ogni serie, e gli operatori formati da un solo glifo, come $\times, \in, \infty, \ldots$ che si possono usare anche con la matematica estesa mediante i font della American Mathematical Society; questo implica, perciò, che *non si deve caricare il file di estensione* `amssymb` *né, meno che mai, il file* `amsfonts`.

I simboli disponibili in tutte le forme e serie sono talmente tanti che in un tentativo di elencare tutti i simboli gestibili con `unicode-math` scrivendo accanto a ciascuno qualche parola di spiegazione, si sono composte circa 80 pagine A4. Siccome non è opportuno replicare le 80 pagine in questo testo, si rinvia alla documentazione del pacchetto `unicode-math` che, oltre alla lista dei simboli, contiene anche alcune indicazioni d'uso per certe macro che "costruiscono" alcuni simboli speciali.

L'unica difficoltà nascosta in quei semplici comandi consiste nel conoscere il nome della famiglia e i nomi dei font particolari. A questo punto bisogna leggere le istruzioni contenute nella documentazione di `fontspec` con il solito comando da linea di comando `texdoc fontspec`. Si scopre allora che questo pacchetto può fare moltissime cose molto specializzate il cui solo elenco qui sarebbe troppo lungo.

Tuttavia il problema persiste; `fontspec` non dice come si può trovare il nome della famiglia, ma solo come usarlo.

Il sistema TeX, già in prima installazione, contiene un piccolo programma da linea di comando che si chiama otfinfo; la sua documentazione si legge mediante il solito comando da linea di comando `texdoc otfinfo`; è opportuno leggere quella documentazione, dalla quale si evince che esso accetta molte opzioni, oltre al nome del file che contiene la descrizione del font di cui fornire le proprietà, ivi incluso il nome della famiglia. Ma se non si specificano le opzioni giuste si rischia di ottenere troppa informazione che coinvolge molte schermate nelle quali ci si può perdere. È meglio essere selettivi.

La sintassi del comando otfinfo nel terminale o prompt dei comandi è la seguente:

```
...$ otfinfo ⟨opzioni⟩ ⟨file⟩
```

dove `...$` è quanto scrive il sistema operativo nel terminale quando aspetta l'introduzione di un comando da parte dell'utente; nelle macchine Windows al posto del segno del dollaro c'è il segno `>`; inoltre ⟨*file*⟩ è il nome di un file contenente le informazioni relative al font che si intende analizzare: si tratta di un indirizzo completo a partire dalla radice del disco; se si volessero le informazioni relative al font Arial sul Mac, dovrei scrivere il comando

```
...$ otfinfo ⟨opzioni⟩ /Library/Fonts/Microsoft/Arial.ttf
```

Già il fatto di dover trovare nel disco il nome del file con il suo percorso a partire dalla radice del disco permette di ottenere qualche informazione sul font, per esempio che questo è stato installato insieme alla suite Microsoft Office™; si vede dall'estensione che si tratta di un file TrueType.

Se come ⟨*opzioni*⟩ specifico solo `-s` il programma mi stampa sullo schermo l'elenco degli alfabeti che contiene; dunque:

```
...-MacBook-Pro: $ otfinfo -s /Library/Fonts/Microsoft/Arial.ttf
arab Arabic
arab.URD Arabic/Urdu
cyrl Cyrillic
grek Greek
hebr Hebrew
latn Latin
latn.LTH Latin/Lithuanian
```

Ecco dunque che veniamo a sapere che con questo font si può comporre nelle lingue che usano l'alfabeto latino, l'arabo, anche nella varietà Urdu, l'alfabeto cirillico; si può scrivere anche in ebraico, in greco e in lituano che usa una particolare varietà dell'alfabeto latino.

Le opzioni disponibili sono le seguenti.

-s mostra gli *script*, gli alfabeti presenti nel font.
-f mostra le varianti che sono presenti nel font e che possono essere selezionate dall'utente; per esempio: creare il maiuscoletto a partire dal maiuscolo; usare stili diversi per il minuscolo e il maiuscolo; spazieggiare le maiuscole; distinguere lo zero 0 dalla O maiuscola barrando lo zero; eccetera.
-z mostra le informazioni disponibili in merito ai corpi ottici.
-p mostra il nome PostScript del font, talvolta necessario per usare comandi di basso livello sottostanti a quelli definiti da *fontspec*.
-a mostra il nome della famiglia.
-v mostra il numero di versione del font.
-i mostra le informazioni generali relative al font: la forma; il nome della famiglia; il nome PostScript del font; la versione; l'identificatore unico secondo lo standard delle ditte produttrici; l'autore; il venditore, almeno il suo URL; il

trade mark in termini verbali; il copyright; la licenza d'uso, almeno il suo URL.

Questa è forse una delle opzioni più interessanti; se la invochiamo per il font Arial, otteniamo quanto segue.

```
...-MacBook-Pro:~ $ otfinfo -i /Library/Fonts/Microsoft/Arial.ttf
Family:              Arial
Subfamily:           Regular
Full name:           Arial
PostScript name:     ArialMT
Version:             Version 5.06
Unique ID:           Monotype:Arial Regular:Version 5.06 (Microsoft)
Designer:            Monotype Type Drawing Office - Robin Nicholas,
                     Patricia Saunders 1982
Manufacturer:        The Monotype Corporation
Trademark:           Arial is a trademark of The Monotype Corporation
                     in the United States and/or other countries.
Copyright:           © 2008 The Monotype Corporation. All Rights Reserved.
                     License Description: You may use this font to display
                     and print content as permitted by the license terms
                     for the product in which this font is included.
                     You may only (i) embed this font in content as
                     permitted by the embedding restrictions included
                     in this font; and (ii) temporarily download this
                     font to a printer or other output device to help print
                     content.
Vendor ID:           TMC
```

Se la invochiamo per il font Latin Modern Math otteniamo quanto segue.

```
...-MacBook-Pro:~ $ otfinfo -i ~/Library/Fonts/latinmodern-math.otf
Family:              Latin Modern Math
Subfamily:           Regular
Full name:           LatinModernMath-Regular
PostScript name:     LatinModernMath-Regular
Preferred family:    Latin Modern Math
Preferred subfamily: Regular
Version:             Version 1.958
Unique ID:           1.958
Trademark:           Please refer to the Copyright section for the font
                     trademark attribution notices.
Copyright:           Copyright 2012 for Latin Modern Math OTF by
                     B. Jackowski, P. Strzelczyk and P. Pianowski
                     (on behalf of TeX users groups). This work is
                     released under the GUST Font License --
                     see http://tug.org/fonts/licenses/GUST-FONT-LICENSE.txt
                     for details.
Vendor ID:           PfEd
```

-g mostra i *nomi* di tutti i glifi contenuti nel file; attenzione ad usare questa opzione perché può scrivere migliaia di righe; se interessa conservare questa lista bisogna reindirizzare l'uscita del programma verso un file; l'operazione può essere differente a seconda del sistema operativo.

-u mostra sia il *nome* sia l'*indirizzo* UNICODE di ogni glifo contenuto nel font; si veda la descrizione dell'opzione precedente.

-t mostra le dimensioni e il nome di ogni tabella UNICODE contenuta nel font.

-T ⟨*tabella*⟩ mostra il contenuto della particolare ⟨*tabella*⟩ OpenType specificata.
--script = ⟨*script*⟩`[`⟨*.lang*⟩`]` usato insieme all'opzione `-f` permette di mostrare le particolarità associate solo all'alfabeto indicato con ⟨*script*⟩, facoltativamente esteso con la specificazione della lingua; per esempio volendo le specifiche del font latino per il lituano bisognerebbe specificare `--script = latn.LTH -f`.
-V mostra l'operato del programma in modo "verboso".
-q ordina al programma di lavorare in modo quieto, senza scrivere informazioni non relative alle proprietà del font.
-h ordina al programma di mostrare un riassunto di quanto esposto qui per esteso.
--version mostra il numero di versione del programma.

Ecco, sembra complicato, ma è semplicissimo, come si è mostrato con i due soli esempi fatti.

Questa lunga spiegazione andrebbe completata con una attenta lettura di *fontspec*; in ogni caso il tutto si riduce a poche cose come mostra i seguente preambolo preso da un altro documento, visto che questo testo è stato composto con pdflatex.

```
\documentclass[12pt,twoside]{toptesi}
\ProvidesFile{toptesi-it.tex}[2020 v....]
...
    \usepackage{fontspec}
    \usepackage{polyglossia}
    \setmainlanguage[babelshorthands]{italian}
    \setotherlanguage[variant=ancient]{greek}

    \setmainfont[Ligatures=TeX]{TeX Gyre Termes}
    \setsansfont[Ligatures=TeX, Scale=MatchLowercase]{TeX Gyre Heros}
    \setmonofont{UM Typewriter}
    \newfontfamily{\greekfont}{GFS Didot}
    \usepackage{amsmath}
    \usepackage{unicode-math}
    \setmathfont{XITS Math}
...
\usepackage{booktabs,array,tabularx,fancyvrb,xcolor,imakeidx}

\usepackage{hyperref}
    \hypersetup{%
        pdfpagemode={UseOutlines},
        bookmarksopen,
        pdfstartview={FitH},
        colorlinks,
        linkcolor={blue},
        citecolor={blue},
        urlcolor={blue}
    }
```

```
\makeindex[intoc,columns=2]
...
```

Si riconoscono in questo preambolo molte delle cose spiegate anche in capitoli precedenti; in particolare si vede come sono state definite le tre famiglie principali per i font e come sono stati associati i font Greci Didot alla lingua greca.

Come si vede queste impostazioni sono un poco più dettagliate rispetto a quelle che si userebbero con il programma pdflatex, ma sono molto più flessibili che non quelle destinate all'uso dei font codificati con un byte; la cosa è del tutto ovvia: i font OpenType permettono un numero molto maggiore di personalizzazioni, per cui anche i comandi per le personalizzazioni sono più dettagliati. Vale la pena osservare che se non si specificano font diversi per le tre famiglie principali, i font usati per impostazione predefinita sono i Latin Modern nella loro forma OpenType.

18.8.2 Font Type 1

Solo pdflatex

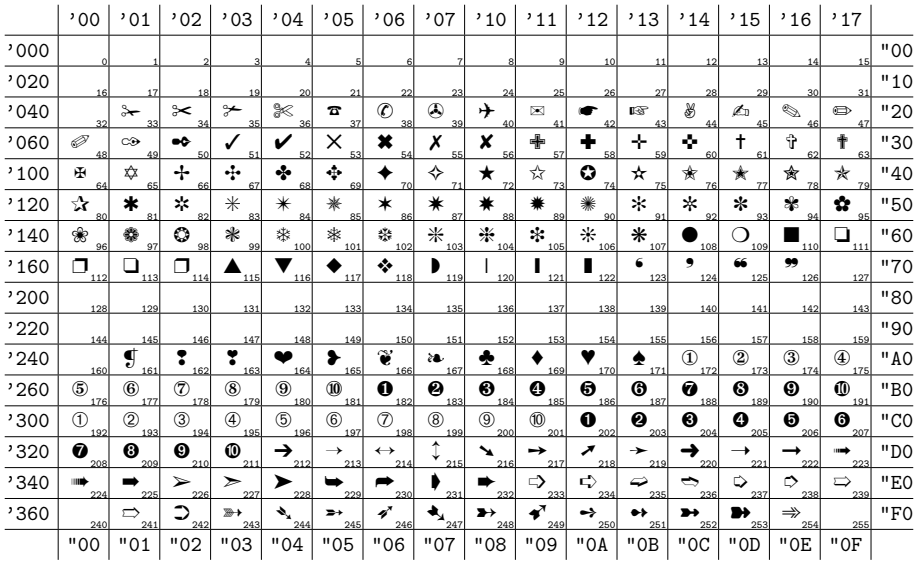
I font Type 1 sono codificati con un byte, e sono i soli font vettoriali che si possono usare con pdflatex. Sebbene i font Type 1 siano usabili anche con i programmi lualatex e xelatex, per servirsi contemporaneamente di questi font e di quelli OpenType bisogna ricorrere a certe astuzie, che possono essere lette nella documentazione di *fontspec*. Qui dunque ci occuperemo solo dell'uso dei font Type 1 mediante il programma pdflatex.

18.8.2.1 Altri font già disponibili

Prima di descrivere la delicata operazione dell'installazione di nuovi font Type 1, vale la pena di descrivere i pacchetti che fanno riferimento a font già disponibili in ogni distribuzione del sistema TeX. Talvolta è necessario scaricare da CTAN ed installare pacchetti già predisposti con tutto il necessario, ma molto spesso i pacchetti fanno parte della dotazione ordinaria del sistema TeX quando ne sia stata fatta una installazione completa.

Fra questi pacchetti si sono già segnalati quelli corrispondenti ai font Times eXtended e Palatino eXtended. L'aggettivo 'extended' (esteso) si riferisce al fatto che le collezioni dei segni usabili sono più ampie (estese) rispetto all'insieme di segni forniti ordinariamente dai font delle collezioni CM, EC e TS arricchite dai segni delle collezioni fornite dall'American Mathematical Society. Il pacchetto *newtx* mette a disposizione due file di estensione: *newtxtext* e *newtxmath* che possono venire usati separatamente. Questa nuova versione corregge alcuni piccoli difetti di accostamento nella composizione della matematica che invece erano presenti con i font Times usati dal pacchetto *txfonts*. Lo stesso è avvenuto con i font Palatino eXtended.

Il documento *Using common PostScript fonts with LaTeX*, leggibile dando da terminale il comando `texdoc psnfss2e`, descrive tutti i font PostScript con i relativi pacchetti e le rispettive opzioni già predisposti per usare i font Palatino (font diversi e con prestazioni inferiori a quelle dei pacchetti *pxfonts* e *newpx*),



Parametri

inclinazione	0,00000	x-height	3,99998 pt
spazio interparola	2,77999 pt	larghezza del quadrato	10,00000 pt
allungamento interparola	3,00000 pt	spazio extra	0,00000 pt
accorciamento interparola	1,00000 pt	corpo nominale	10,00000 pt

Figura 18.8: I *dingbats* creati da Hermann Zapf

i Times (anche questi con prestazioni inferiori a quelle fornite dai pacchetti *txfonts* e *newtx*) Helvetica, Avant Garde, Courier, Zapf Chancery, Bookman, New Century Schoolbook, Charter, Utopia. Il documento descrive anche i limiti di ogni pacchetto e segnala i pacchetti obsoleti da *non* usare, se non per comporre vecchi documenti che ricorrevano ad alcuni di questi font.

Il documento segnala anche alcuni pacchetti, per esempio *Pifonts*, da caricare per disporre dei simboli e dei 'dingbats' forniti dalle polizze che offrono solo questo genere di segni. Diventa così abbastanza semplice disporre di tutti quei *pittogrammi* usati nel mondo occidentale per segnalare luoghi specifici (telefono, luoghi di culto, aeroporti, ...), delle manine che indicano e attraggono l'attenzione su un oggetto o una parola, motivi di decorazione, cifre da 0 a 10 variamente stilizzate, molti tipi di frecce, eccetera. La figura 18.8 presenta una di queste polizze di segni speciali.

Va segnalato che con alcuni di questi font, e per altri non indicati, il pacchetto *mathdesign* consente di associare i font matematici e le collezioni di simboli in modo che molte delle limitazioni e degli inconvenienti segnalati dal documento `psnfss2e.pdf` possono venire eliminati. Questo pacchetto consente di estendere i font Utopia, Charter e Garamond; esso consente quindi di personalizzare forte-

mente l'aspetto grafico dei propri lavori di composizione, senza però rinunciare a nessuna delle prerogative di LaTeX. Chi scrive, però, ha raramente usato questi altri font, non perché egli li consideri inadatti, ma perché si è decisamente trovato meglio con i font Latin Modern, anche se i suoi scritti sembrano 'monòtoni' per mancanza di varietà di font.

Chi scrive ha usato talvolta l'uno o l'altro dei pacchetti *fourier* e *kpfonts*; il primo offre una installazione completa per il testo e la matematica, compresi i simboli della American Mathematical Society, e molti altri simboli e altri alfabeti. Si basa sui font Utopia, molto belli e leggibili, visto che a pari corpo hanno un occhio più grande dei Latin Modern. Analogamente il secondo pacchetto offre pressappoco le stesse prestazioni del primo, ma usando come font principale un'ottima approssimazione del font Palladio; chi scrive apprezza molto entrambi questi pacchetti.

In questo testo sono stati usati tra gli altri i font della famiglia Text Companion vettoriali; questa collezione di font si affianca ai font di default per mettere a disposizione un altro centinaio di lettere e segni analfabetici molto utili in diverse circostanze.

Il sistema TeX contiene già preinstallati i font della collezione `CM-super`, completa di file metrici, di font virtuali, di font vettoriali e di tutte le varianti con le codifiche *T1* (caratteri latini), *T2* (caratteri cirillici) e *TS1* (simboli della collezione Text Companion); grazie ai font virtuali, condivide quei segni che sono identici fra i caratteri latini e cirillici. Essi figurano già nelle mappe relative ai caratteri vettoriali e non è necessario preoccupassi della loro installazione.

Sono preinstallati anche i font Latin Modern (LM) che ora sono distribuiti anche nella forma OpenType e sono predefiniti quando si usano xelatex e lualatex; essi offrono una migliore collezione di segni vettoriali, apparentemente migliori dei CM-super secondo molti utenti. Certamente, se non si ha bisogno di usare il cirillico, la collezione Latin Modern è molto più parca per quel che riguarda lo spazio sul disco.

I font della collezione Latin Modern comprendono anche i file di ricodifica conformi alle norme Adobe per i font PostScript e permettono di servirsi di codifiche diverse dalle solite. È possibile usarli anche con codifiche diverse da *T1*, ma non c'è bisogno di fare nulla per la loro installazione.

Tra le codifiche con cui possono essere usati i font Latin Modern, c'è anche la codifica *LY1*; è una codifica interessante, perché consente di evitare il caricamento dei font Text Companion, a meno che non si debbano usare simboli molto particolari; per esempio per usare il simbolo dell'euro il comando `\texteuro` richiederebbe l'uso di font Text Companion se per il testo si usa la codifica T1, mentre se si usa la codifica LY1, non ce n'è bisogno; lo stesso vale per altri pochi segni; è meglio verificare sulla tabella 18.13 per vedere che cosa è disponibile e che cosa manca.

Ricordo quanto già segnalato: gli ultimi aggiornamenti (vedi il capitolo 30) del nucleodi LaTeX non richiedono più l'invocazione dell'apposito pacchetto per usare il Text Companion font.

	’00	’01	’02	’03	’04	’05	’06	’07	’10	’11	’12	’13	’14	’15	’16	’17	
’000	0	€ 1	2	3	⁄ 4	˙ 5	˝ 6	˛ 7	fl 8	9	10	ff 11	fi 12	13	ffi 14	ffl 15	"00
’020	ı 16	J 17	` 18	´ 19	ˇ 20	˘ 21	¯ 22	° 23	¸ 24	ß 25	æ 26	œ 27	ø 28	Æ 29	Œ 30	Ø 31	"10
’040	32	! 33	" 34	# 35	$ 36	% 37	& 38	’ 39	(40	) 41	* 42	+ 43	, 44	- 45	. 46	/ 47	"20
’060	0 48	1 49	2 50	3 51	4 52	5 53	6 54	7 55	8 56	9 57	: 58	; 59	< 60	= 61	> 62	? 63	"30
’100	@ 64	A 65	B 66	C 67	D 68	E 69	F 70	G 71	H 72	I 73	J 74	K 75	L 76	M 77	N 78	O 79	"40
’120	P 80	Q 81	R 82	S 83	T 84	U 85	V 86	W 87	X 88	Y 89	Z 90	[91	\ 92	] 93	^ 94	_ 95	"50
’140	‘ 96	a 97	b 98	c 99	d 100	e 101	f 102	g 103	h 104	i 105	j 106	k 107	l 108	m 109	n 110	o 111	"60
’160	p 112	q 113	r 114	s 115	t 116	u 117	v 118	w 119	x 120	y 121	z 122	{ 123	| 124	} 125	~ 126	¨ 127	"70
’200	Ł 128	' 129	‚ 130	f 131	„ 132	... 133	† 134	‡ 135	ˆ 136	‰ 137	Š 138	‹ 139	Œ 140	Ž 141	ˆ 142	— 143	"80
’220	ł 144	‘ 145	’ 146	“ 147	” 148	• 149	– 150	— 151	˜ 152	TM 153	š 154	› 155	œ 156	ž 157	~ 158	Ÿ 159	"90
’240	160	¡ 161	¢ 162	£ 163	¤ 164	¥ 165	¦ 166	§ 167	¨ 168	© 169	ª 170	« 171	¬ 172	- 173	® 174	¯ 175	"A0
’260	° 176	± 177	2 178	3 179	´ 180	µ 181	¶ 182	· 183	¸ 184	1 185	º 186	» 187	¼ 188	½ 189	¾ 190	¿ 191	"B0
’300	À 192	Á 193	Â 194	Ã 195	Ä 196	Å 197	Æ 198	Ç 199	È 200	É 201	Ê 202	Ë 203	Ì 204	Í 205	Î 206	Ï 207	"C0
’320	Đ 208	Ñ 209	Ò 210	Ó 211	Ô 212	Õ 213	Ö 214	× 215	Ø 216	Ù 217	Ú 218	Û 219	Ü 220	Ý 221	Þ 222	ß 223	"D0
’340	à 224	á 225	â 226	ã 227	ä 228	å 229	æ 230	ç 231	è 232	é 233	ê 234	ë 235	ì 236	í 237	î 238	ï 239	"E0
’360	ð 240	ñ 241	ò 242	ó 243	ô 244	õ 245	ö 246	÷ 247	ø 248	ù 249	ú 250	û 251	ü 252	ý 253	þ 254	ÿ 255	"F0
	"00	"01	"02	"03	"04	"05	"06	"07	"08	"09	"0A	"0B	"0C	"0D	"0E	"0F	

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz

Parametri

inclinazione	0,00000	x-height	4,30549 pt
spazio interparola	3,33332 pt	larghezza del quadrato	10,00000 pt
allungamento interparola	1,66665 pt	spazio extra	1,11111 pt
accorciamento interparola	1,11111 pt	corpo nominale	10,00000 pt
alfabeto maiuscolo	186,18060 pt	alfabeto minuscolo	127,58364 pt

Tabella 18.13: Il font latino con codifica LY1

18.8.2.2 Installazione di altri font vettoriali

Se si va ad aprire il file `pdftex.map` si scopre che contiene 12 445 righe (almeno sul calcolatore di chi scrive, che in aggiunta ai font normalmente distribuiti con l’installazione TEX Live completa, ha aggiunto una ventina di font personali) fra le quali poche righe di commento; si può stimare, visto che ogni riga si riferisce ad un diverso font, che su questo calcolatore la distribuzione TEX Live completa mette a disposizione circa 12 000 diversi font vettoriali; sembra impossibile che con tale varietà di scelta si debbano installare altri font, ma succede.

Qui si esporranno alcune situazioni che si presentano effettivamente, non molto spesso, ma si presentano. Installare nuovi font vettoriali è difficile specialmente per le numerose piccole attenzioni che bisogna avere ad ogni passo della procedura. Nel seguito mi riferirò specialmente alla distribuzione TEX Live, ma non trascurerò MiKTEX, visto che alcune operazioni da compiere con quest’ultimo sono diverse.

Vale la pena premettere la scaletta delle operazioni da compiere per installare ciascun nuovo font vettoriale; ogni passo verrà illustrato nei paragrafi successivi.

1. Creare i file metrici `.tfm` e, se occorre, il file virtuali `.vf`:
   - per font Type 1 generati con FontForge;
   - per font Type 1 generati a partire da font creati con METAFONT;
   - per i font Type 1 acquistati o scaricati dalla Rete;
   - per i font TrueType;
   - per i font OpenType.
2. Creare i file di descrizione dei font `.fd` e magari anche il pacchetto di estensione `.sty`.
3. Creare i file di mappa `.map`:
   - con TeX Live;
   - con MiKTeX.
4. Aggiornare i file generali delle mappe:
   - per un solo utente;
   - per tutti gli utenti della macchina
     - con TeX Live;
     - con MiKTeX.
5. Verificare di disporre dei programmi per eseguire i compiti precedenti.

La lista sembra lunga, e lo è, ma se si ha la pazienza di seguire con attenzione i vari passi nell'ordine in cui sono esposti, non dovrebbe essere difficile installare nuovi font vettoriali. In ogni caso procederò per esempi; quindi è possibile che alcuni punti della procedura non vengano esposti; in questi casi il lettore è indirizzato verso la documentazione prevalentemente contenuta in quella di TeX Live (dare il comando `texdoc texlive` dal terminale), oltre che nel messaggio di aiuto del programma `updmap` (dare il comando `updmap -help` dal terminale); oppure consultando la documentazione di MiKTeX.

Che cosa siano alcuni dei file nominati nella lista, apparirà chiaro da quanto esposto qui di seguito.

18.8.2.3 Operazioni preliminari

Nei paragrafi che seguono si parlerà di diversi tipi di file; è opportuno che ogni tipo di file risieda in una cartella opportuna, possibilmente innestata in una struttura conforme al TDS (TeX Directory System); l'installazione di TeX Live già prevede questa struttura. MiKTeX vi si avvicina abbastanza, ma sembra che la sua struttura di cartelle non sia completamente conforme alle direttive TDS. Questa struttura è descritta con abbastanza dettaglio nella documentazione di TeX Live, leggibile con `texdoc texlive`. Partendo dalla radice dell'unico albero o dall'insieme di alberi della distribuzione installata, dovrebbe contenere diverse ramificazioni:

- `fonts/tfm/` destinata a contenere i file metrici;
- `fonts/vf/` destinata a contenere il file dei font virtuali;

- `fonts/type1/` destinata a contenere i file dei font Type 1;
- `fonts/truetype/` destinata a contenere i file dei font TrueType
- `fonts/opentype/` destinata a contenere i file dei font OpenType;
- `fonts/map/` destinata a contenere i file delle mappe;
- `fonts/enc/` destinata a contenere i file di codifica dei font virtuali;
- `fonts/pk/` destinata a contenere i file dei font bitmapped;
- `web2c/` destinata a contenere i file di configurazione delle mappe dei font;
- `fonts/source/` destinata a contenere i file sorgente per il programma METAFONT;
- `tex/latex/` destinata a contenere i file di descrizione dei font e, se esistono, i file di estensione per l'uso dei corrispondenti font.

Spesso bisogna replicare questa struttura nell'albero personale e, se c'è, anche nell'albero locale; questo solitamente comincia con una cartella che si chiama `texmf-local`. Invece l'albero personale è una cosa personale dell'utente e non viene creato con l'installazione del sistema TEX; tocca all'utente crearselo con la struttura indicata. Nelle macchine Linux generalmente questo albero personale è radicato in `~/texmf`; nelle macchine Mac esso va radicato in `~/Library/texmf/`; nelle macchine Windows dipende dalla versione: sulle macchine WinVista e successive potrebbe essere radicato in `C:\Users\`⟨*utente*⟩`\texmf`.

In linea generale quello che si installa e non fa parte in senso stretto della distribuzione TEX Live o MiKTEX è bene che sia installato nell'albero personale o, in TEX Live, nell'albero locale. Ciò che viene messo nell'albero locale richiede che sia stato creato e che venga aggiornato il database dei nomi dei file; nell'albero personale questo aggiornamento non è necessario con TEX Live. Il motivo di questa preferenza è che gli alberi pertinenti alla distribuzione in senso stretto vengono riscritti ad ogni installazione di una nuova versione della propria distribuzione del sistema TEX; a parte gli eventuali privilegi per scrivere nelle cartelle della distribuzione, mettere in queste cartelle qualcosa che non fa parte in senso stretto della distribuzione vuol dire perderlo completamente ad ogni aggiornamento di una nuova versione. Sarebbe un "infortunio" particolarmente sgradevole.

Con TEX Live ciò che sta nell'albero locale è accessibile a tutti gli utenti della macchina *in lettura*; quindi per *scrivere* qualcosa nelle cartelle di questo albero bisogna avere le prerogative dell'amministratore. Invece nell'albero personale scrive e legge solo il proprietario. Sulle macchine Windows il concetto di amministratore è ancora abbastanza vago, ma sta diventando un po' più definito con le ultime versioni del sistema operativo, da Vista in poi. Da molte versioni gli utenti sono divisi in amministratori, utenti con permessi estesi, utenti normali, ospiti, eccetera, ma di fatto le differenze per quel che riguarda il sistema TEX sono abbastanza modeste. Con i sistemi operativi di tipo UNIX, il concetto di amministratore è sempre stato molto preciso come lo sono i modi di accesso (lettura, scrittura, esecuzione) per i singoli file.

Nel seguito si supporrà che ogni tipo di file sia collocato nella cartella che gli pertiene, possibilmente con un nome fortemente mnemonico, collocata lungo

la ramificazione giusta dei vari alberi descritti sopra, e non lasciato in cartelle qualsiasi. I programmi del sistema TeX sanno dove trovare i file sui quali devono operare solo se sono nelle ramificazioni giuste.

18.8.3 Installare un font Type 1 creato con FontForge

È capitato a chi scrive di creare font con il programma FontForge; questo è un programma gratuito disponibile per tutte e tre le piattaforme principali che conviene installare anche per altri scopi; si tratta di un editor di font vettoriali con il quale è possibile creare font partendo da zero, come anche convertire in formato Type 1 font bitmapped generati con METAFONT; riesce anche a creare font vettoriali partendo da immagini bitmapped di glifi fotografati o copiati da una schermata. In uscita può produrre font vettoriali in diversi formati, il più comune dei quali è appunto il formato Type 1, ma è in grado di creare anche font OpenType.

Lo scrivente ha dovuto creare il logo della Università delle Tre Età e lo ha fatto con FontForge creando direttamente il font `uni3.pfb` e il corrispondente file `uni3.tfm`. Questo font non ha una codifica particolare perché contiene solo un glifo nella posizione corrispondente alla lettera 'U'; nel margine vi appare il logo ingrandito e colorato di verde come appare nei documenti di quell'ente. Per questo motivo per scrivere questo glifo non occorre né un file di descrizione del font (benché non sia vietato) né un file contenente qualche definizione per comporre questo solo glifo; basta avere definito nel preambolo o in un file di macro personali quanto segue:

```
\newfont{\unitre}{uni3}
\newcommand*\LogoUniTre{{\unitre U}}
```

Il primo comando non è altro che un comando del vecchio LaTeX 209 che, dopo aver controllato che la macro sia definibile, associa il nome della macro `\unitre` al font appena creato; il secondo comando permette di comporre il logo della Università delle Tre Età.

Ma questo non basta; perché pdfLaTeX possa usare questo font Type 1 bisogna dirgli che il font esiste e quale font vero corrisponde al nome del file metrico. Questo si fa in due tempi; prima si scrive un file `.map` che descrive quella associazione, per esempio `uni3.map`; poi si inserisce questo file di mappa nel file generale con il quale pdfLaTeX accede ai font realmente esistenti. Il file di mappa per questo font, con il nome, per esempio, `uni3.map`[9] ha questo contenuto:

```
uni3␣uni3␣<uni3.pfb
```

La prima istanza del nome 'uni3' si riferisce al nome senza estensione del file metrico; la seconda istanza si riferisce al nome del font contenuto dentro il file

[9] Il nome può essere scelto a piacere; basta usarlo coerentemente. Ma il contenuto del file di mappa qui indicato può anche essere inserito in un file di mappa generico che raccoglie le informazioni necessarie anche per altri font.

`.pfb`, nome che si può agevolmente leggere per mezzo di FontForge; il terzo elemento preceduto dal segno `<` è il nome per esteso, completo di estensione del file vettoriale che contiene la descrizione grafica dei glifi contenuti nel font.

Per il secondo passo, quello di inserire questa informazione nei file generali di mappa, di veda più avanti il paragrafo 18.8.8. Dopo aver eseguito questo secondo passo, pdflatex (e gli altri programmi che usano i font vettoriali: dvips e dvipdfmx) possono usare il font descritto in questa sezione.

18.8.4 Installare un font Type 1 creato con METAFONT

Supponiamo di disporre dei file sorgente `mf` per creare dei font con METAFONT; che questi file facciano parte della distribuzione (come, per esempio, i file della collezione dei font EC), oppure che si siano caricati dalla rete, oppure che l'utente abbia creato da solo per disegnare certi suoi font non fa nessuna differenza, purché questi file siano nella ramificazione di cartelle giusta.

Eseguendo METAFONT vengono generati i file `.tfm` e i file delle bitmap dei font; questi sono file con l'estensione formata agglutinando la densità di pixel al pollice con la desinenza `pk`; quindi il font bitmapped cmr10 creato per lavorare con una stampante da 300 dpi ha il suo file che si chiama `cmr10.300pk`. Questi file 'pk' non rappresentano font vettoriali, e se servono a qualcosa, servono solo se si desidera comporre per ottenere il file composto in formato DVI senza volerlo trasformare in altri formati; se lo si trasformasse, i programmi di trasformazione userebbero i font vettoriali se ne esistono con lo stesso nome.

Bisogna trasformare il font bitmapped appena creato con METAFONT in un font vettoriale; a questo scopo si può usare lo script python mftrace. Innanzi tutto se non lo si è già fatto bisogna installarselo dalla rete; inoltre, mentre l'interprete python è già presente sulle macchine UNIX (Linux e Mac) questo interprete generalmente non fa parte dei sistemi operativi Windows, quindi bisogna installarselo; in alternativa è possibile installarsi il sistema CygWin che forma sulle macchine Windows un ambiente virtuale che simula una macchina UNIX; in questo caso il programma mftrace è bene che sia installato nella "macchina virtuale" CygWin, nella quale si installeranno anche gli altri programmi indicati in questo paragrafo.

Supponendo che il file da trasformare si chiami `miofont.mf`, si lancia poi il programma mftrace con il comando

```
mftrace -f PFB -e encfile --potrace --simplify miofont
```

dove `-f PFB` specifica il formato di uscita ed è obbligatorio se si vuole un formato diverso da PFA che è il formato di uscita predefinito; `-e encfile` serve per specificare facoltativamente il nome del file che definisce l'ordine nel quale sono elencati i vari glifi all'interno del file di uscita; questi file si trovano nell'apposita ramificazione indicata precedentemente; `--potrace` specifica che si vuole usare il programma potrace per tracciare il contorno di ogni glifo; `--simplify` serve per semplificare la descrizione del contorno espressa mediante l'elenco dei nodi e dei punti guida delle spline di Bézier, visto che l'operazione di tracciamento

potrebbe avere indicato troppi nodi e punti di controllo in relazione ad una certa approssimazione consentita; ovviamente `miofont` è il nome del font che si vuole trasformare in vettoriale ma di cui si dispone solo della versione per METAFONT. Naturalmente si possono specificare altre opzioni; le istruzioni per eseguire questo programma si trovano soltanto nello stesso sito da cui lo si può scaricare `http://lilypond.org/mftrace/`.

A grandi linee mftrace opera così: genera nuovamente il file `pk` del font in questione con un'altissima densità di pixel al pollice, come se dovesse creare un font bitmapped per caratteri cubitali; si capisce che maggiore è la dimensione dell'immagine bitmapped di ogni glifo, minore è l'influenza della seghettatura del suo contorno. Traccia i contorni dei vari glifi mediante spline di Bézier del terzo ordine e memorizza il risultato in un file vettoriale intermedio che abbia il formato giusto per essere dato in pasto al programma FontForge (che quindi deve già essere installato); FontForge provvede a semplificare l'insieme dei nodi e dei punti di controllo dei contorni dei glifi e genera i font in formato `.pfb`. Alla conclusione mftrace cancella tutti i file ausiliari su cui ha lavorato, lasciando nella cartella di lavoro solo il file `.pfb`.

Il file `.tfm` non è stato toccato, quindi si trova già al posto giusto, ma bisogna ancora spostare il file `.pfb` nell'opportuna cartella della ramificazione giusta.

Non resta che creare il file di mappa e procedere, quindi, come spiegato alla fine del paragrafo 18.8.3.

18.8.5 Installare font Type 1

I font Type 1 che si possono acquistare o scaricare dalla rete sono dotati di un file `.pfb` (raramente di un file in formato ASCII `.pfa`) e di un file metrico nel formato `.afm` (Adobe Font Metrics). Il file `.pfa` nel formato leggibile dagli uomini è piuttosto ingombrante; è meglio trasformarlo nel formato `.pfb`; per questo scopo è meglio affidarsi al programma FontForge.

Tuttavia non basta; i glifi all'interno del file `.pfb` potrebbero essere disposti in un ordine diverso da come se lo aspetta il sistema TEX; per ovviare a questo e per ottenere il file metrico per il sistema TEX e, facoltativamente, il font virtuale, basta eseguire il programma afm2tfm specificandogli le opzioni giuste e il file di codifica giusto; per i dettagli bisogna consultare la documentazione; con `texdoc afm2tfm` si ottiene una documentazione scarna e obsoleta, mentre con il comando `afm2tfm -help` si ottiene una documentazione succinta ma sufficiente allo scopo. Si noti che se si vuole generare anche il font virtuale, questo esce in formato leggibile `.vpl` (Virtual Property List); per convertirlo in formato binario `.vf`, quello che viene effettivamente usato dai programmi di trasformazione o di composizione, bisogna trasformarlo ancora nel formato desiderato mediante il programmino vptovf.

Non resta che creare il file di mappa come indicato alla fine del paragrafo 18.8.3.

Questa semplice sequenza di operazioni che può includere anche il trasferimento dei vari file nelle cartelle giuste della diramazione giusta, può venire

eseguita mediante uno script bash per le macchine UNIX o bat per le macchine Windows. Alternativamente si possono usare programmi o script come fontinst (un po' vecchiotto) o autoinst alla cui documentazione si rinvia il lettore: `texdoc fontinst` e `texdoc autoinst`. Questi programmi, specialmente il secondo, possono provvedere anche a spostare i vari file nelle cartelle giuste di una struttura TDS.

18.8.6 Installare font TrueType

Per i font TrueType (codificati con un solo byte) le cose si sviluppano in modo del tutto simile a quello seguito per i font Type 1: si esegue il programmino ttf2tfm che estrae il font metrico per il sistema TEX direttamente dai font TrueType, non da un file separato, e crea anche il file virtuale.

Bisogna ancora creare il file di mappa come indicato alla fine del paragrafo 18.8.3; un'unica avvertenza: nel file di mappa bisogna indicare nel nome del file reale l'estensione `.ttf` invece dell'estensione `.pfb`; sembra ovvio, ma è meglio ricordarlo.

18.8.7 Installare font OpenType

In linea di principio l'installazione di un font OpenType, alcuni dei quali hanno l'estensione `.ttf` come i font TrueType ma hanno i glifi codificati con più di un byte, da usare con pdflatex non differisce granché da quella di un font TrueType; in pratica la cosa è molto più complessa, perché i font OpenType contengono moltissimi glifi, e dispongono di proprietà specifiche per usare certi glifi al posto di altri; per esempio, spesso dispongono di cifre maiuscole (tutte alte uguali e come le lettere maiuscole) e di cifre minuscole (le cifre di altezze diverse, alcune alte come le lettere minuscole, alcune con ascendenti, altre con discendenti; si confronti infatti 0123456789 con 0123456789).

L'operazione si esegue in modo relativamente semplice mediante il programma autoinst che consente, almeno con una installazione conforme alle direttive TDS, di creare i file virtuali necessari con le transcodifiche necessarie, insieme ai file metrici per l'uso con pdflatex, spostandoli tutti nelle cartelle giuste; crea anche i file di mappa. La documentazione leggibile con `texdoc autoinst` è abbastanza chiara, ma bisogna avere familiarità con diversi termini tecnico-informatici.

Riuscendo però a compiere questa operazione, non resta che creare i file generali di mappa, come viene descritto qui di seguito.

18.8.8 Aggiornamento dei file generali di mappa

I file generali di mappa sono le raccolte di tutti i file di mappa per tutti i font che l'utente vuole usare; sono scritti nei linguaggi che i programmi pdftex, dvips e dvipdfm sanno interpretare; sono rispettivamente i file `pdftex.map`, `psfonts.map`, `dvipdfm.map`, oltre ad altri file per altri programmi di trasformazione.

Senza questi file i programmi di composizione non possono riconoscere la disponibilità dei font vettoriali, quindi non possono lavorare o non possono produrre risultati corretti.

18.8.8.1 TeX Live

Dalla versione 2012 di TeX Live le modalità per generare le mappe generali sono cambiate; in un certo senso sono più semplici di quanto non lo fossero con le versioni precedenti.

In pratica consistono nell'eseguire il programma updmap-sys oppure il programma updmap. Il secondo genera i file generali di mappa per l'uso del solo utente che dà il comando, e memorizza i file ottenuti nell'albero personale. Il primo comando, che richiede i privilegi di amministratore, genera i file generali di mappa nell'albero locale che è accessibile a tutti gli utenti della macchina in sola lettura; da questo deriva che per eseguire il programma, l'utente deve acquisire i privilegi dell'amministratore; sulle macchine Windows dovrebbe eseguire il login con lo username di un utente che abbia i privilegi dell'amministratore, se per caso il suo username non è già configurato per disporre di tali privilegi; questo succede spessissimo, visto che la maggior parte dei personal computer sono installati dall'utente che l'ha comprato e che in quanto tale è il primo utente, quindi l'amministratore; coloro che, dopo l'acquisto, fanno eseguire il primo avviamento da una terza persona, generalmente non dispongono dei privilegi dell'amministratore; stranamente sono molto numerose le persone che si fanno fare questa operazione da terzi e ignorano che non possono più amministrare il loro PC. In questo caso saranno "costretti" a creare i file generali di mappa solo per se stessi; probabilmente non noteranno nessuna differenza se sono gli unici utenti del loro PC. Ma una differenza c'è; infatti ogni nuova installazione e ogni aggiornamento di TeX Live non aggiornano mai l'albero personale, e quindi non vengono ricreati i file di mappa aggiornati e l'utente rimane spiazzato, perché non riesce più ad usare i suoi font personali; basta che si ricordi di eseguire updmap e si ritrova il proprio sistema in grado di usare i font personali; ma bisogna riconoscere che per lui/lei è una seccatura; la soluzione è quella di essere amministratori del proprio PC, senza ricorrere a terze persone.

Ma su altre macchine, dove il concetto di amministratore è più definito, è meglio creare i file generali di mappa come amministratore rendendo quindi accessibili questi file, in sola lettura, a chiunque abbia il diritto di usare la macchina.

Per acquisire i diritti dell'amministratore, o del *super user*, che è quasi lo stesso, l'utente deve scrivere il comando `sudo` prima di `updmap-sys`:

```
sudo updmap-sys
```

Dato questo comando, il sistema operativo chiede di immettere la parola chiave dell'amministratore, e se è corretta, esegue il successivo comando.

Tutto ciò è semplicissimo, ma come fa updmap-sys a sapere quali file di mappa deve usare? Ci sono sparsi nei vari rami degli alberi del sistema TeX e nell'albero

personale, diversi file `updmap.cfg`; il programma `updmap-sys` usa tutti questi file di configurazione, o meglio tutti quelli che il programma della distribuzione TEX Live `kpsewhich` riesce a trovare; se si dà questo comando, sul mio Mac in questo momento, nella forma:

```
kpsewhich --all updmap.cfg
```

esso risponde:

```
~/Library/texmf/web2c/updmap.cfg
/usr/local/texlive/texmf-local/web2c/updmap.cfg
/usr/local/texlive/2019/texmf-dist/web2c/updmap.cfg
```

e questi sono i tre file di configurazione che `updmap-sys` concatena e usa tutte le informazioni di ciascun font che sono nominati all'interno di ciascun file. In particolare i tre file di configurazione indicati vengono letti e concatenati nell'ordine in cui sono mostrati sopra. Quindi prima quello mostrato nell'albero personale (HOME), poi quello dell'albero locale (LOCAL), poi quello in cui si trova la maggior parte della distribuzione TEX Live (DIST). Se esaminando questi file di configurazione un file di mappa fosse ripetuto più volte, il programma conserva solo la prima istanza; analogamente se un font fosse ripetuto più volte in diversi file vengono usate solo le informazioni della prima istanza.

Il file di configurazione nell'albero DIST contiene diverse impostazioni generali e la maggior parte di tutte le mappe della distribuzione; il file di configurazione dell'albero LOCAL, invece, contiene le informazioni sulle mappe dei font aggiunti da chi amministra la macchina. In questo calcolatore il file locale di configurazione contiene solo la riga

```
Map local.map
```

A sua volta la mappa `local.map` che si trova nella diramazione `fonts/map/` dell'albero LOCAL contiene qualcosa come:

```
NebioloGreek          NebioloGreek          <NebioloGreek.pfb
athanasius3           athanasius3           <athanasius3.pfb
athanasius4           athanasius4           <athanasius4.pfb
athanasius5           athanasius5           <athanasius5.pfb
athanasius6           athanasius6           <athanasius6.pfb
cbbohairicmedium      cbbohairicmedium      <cbbohairicmedium.pfb
...                   ...                   ...
uni3                  uni3                  <uni3.pfb
```

Ecco quindi il meccanismo vero: per i font installati localmente, in aggiunta a quelli della distribuzione di TEX Live, basta indicarne le specifiche come illustrato alla fine del paragrafo 18.8.3 o in un file singolo o un file collettivo, dal nome arbitrario (qui chiamato `local.map`), e questo o questi file di mappa per font particolari usati solo localmente devono venire trascritti con il prefisso `Map␣` nel file di configurazione `updmap.cfg` dell'albero LOCAL. I file di configurazione presenti negli altri alberi della distribuzione TEX Live non devono venire toccati per nessun motivo.

18.8.8.2 MiKTeX

Per MiKTeX le cose sembrano più semplici, ma sono anche loro abbastanza delicate, perché nella distribuzione MiKTeX per piattaforme Windows non esiste l'albero locale (non si confonda l'albero locale con l'albero personale) e non è ben chiaro chi sia l'amministratore; ciò premesso, creato un file, per esempio `local.map` come indicato sopra per TeX Live, ma memorizzato nell'albero personale nel ramo `texmf/web2c/`, dopo aver aggiornato il database dei nomi dei file, l'utente può dare il comando:

```
initexmf --edit-config-file updmap
```

che consente di editare il file `updmap.cfg` aggiungendoci in fondo la riga:

```
Map local.map
```

Salvato il file, si dà poi il comando:

```
initexmf --makemaps
```

che aggiorna i file generali di mappa.

Chi scrive da diversi anni non dispone più di una versione di MiKTeX aggiornata, quindi non sa se la prima delle due operazioni sia fragile rispetto all'installazione di una nuova versione di MiKTeX; è possibile; ma da un lato gli upgrade di singoli pacchetti in MiKTeX non implicano la riscrittura di tutto l'albero della sua distribuzione, bensì solo delle cartelle coinvolte da quei pacchetti; dall'altro le nuove versioni di MiKTeX non si succedono annualmente come le nuove versioni di TeX Live. Quindi il problema della sovrascrittura, ammesso che si ponga, è molto dilazionato nel tempo.

18.8.9 I file di descrizione dei font

I file di descrizione dei font, con estensione `.fd` non sono obbligatori, ma sono utili. Si è già visto nel paragrafo 18.8.3 e si è commentato a questo proposito che in quel caso il file di descrizione del font, benché non inutile, non era veramente indispensabile. Tuttavia per casi più complessi è quanto mai opportuno disporre di un file di descrizione di una collezione di font, altrimenti per usarli bisogna ricorrere a comandi nativi o a comandi di basso livello.

Innanzi tutto si richiama il fatto che la selezione dei font con il New Font Selection Scheme (NFSS) che ha rivoluzionato LaTeX 2_ε rispetto al precedente LaTeX 209, ma che ora, dopo più di venti anni di esistenza, non sembra più tanto nuovo, ha definito per ogni font 6 caratteristiche che possono venire selezionate le une indipendentemente dalle altre (vedi però anche il capitolo 30).

1. La codifica; ne abbiamo già descritte varie con le loro funzioni e caratteristiche.

2. La famiglia: per ogni collezione di font abbiamo visto che si possono avere almeno tre famiglie, quella dei font con grazie, quella dei font senza grazie e quella dei font monospaziati; alcune collezioni di font possono avere anche altre famiglie.
3. La serie: abbiamo visto che ogni font può essere dotato di glifi più o meno "neri" e più o meno evidenziati, per cui disponiamo sempre della serie media e della serie nera estesa, ma potremmo avere la serie condensata, condensata nera, nera senza estensione, eccetera.
4. La forma: abbiamo già descritto la forma diritta, quella inclinata, quella corsiva, quella corsiva diritta, quella maiuscoletta, ma alcune collezioni di font contengono anche altre forme.
5. Il corpo: abbiamo già spiegato che cosa sia il corpo e sappiamo che con il sistema TEX abbiamo la possibilità di usare nello stesso documento corpi piccolissimi, come quello di 5 pt, e corpi piuttosto grandi come quello di 25 pt; in certi casi si possono avere anche corpi decisamente maggiori e la classe *memoir* può addirittura comporre documenti (poster) usando un corpo 'normale' da 60 pt.
6. Lo scartamento: sappiamo che tradizionalmente lo scartamento (erroneamente chiamato anche interlinea), cioè la distanza fra due linee di base successive, è del 20% maggiore del corpo, ma in certe circostanze è opportuno aumentare o diminuire questo scartamento per ottenere effetti speciali. In certi casi lo scartamento potrebbe essere anche leggermente minore del corpo.

Come fa il programma di composizione, nel nostro caso pdflatex, a sapere di quali caratteristiche è dotato il font corrente? Oppure, come fa pdflatex a sapere di quali caratteristiche disponiamo? Lo fa leggendo i file di descrizione dei font.

Esiste un file distinto per ciascuna codifica e ciascuna famiglia di ogni collezione di font. Anzi, il nome del file, con estensione `.fd`, ha il nome in lettere minuscole formato agglutinando la sigla della codifica e il nome della famiglia; qui, in questo capoverso, si sta componendo con la codifica *T1* e con la famiglia dei font con grazie Latin Modern, la cui sigla è `lmr`, per cui le informazioni sui font disponibili con questa codifica e questa famiglia sono contenuti nel file `t1lmr.fd`. Vale la pena vedere che cosa contiene questo file, ma vale anche la pena segnalare subito che i dettagli per l'esatta creazione di questo genere di file sono contenuti nella guida sui font leggibile dal terminale attraverso il comando `texdoc fntguide`. Comunque il file in questione[10] è il seguente.

```
% This file belongs to the Latin Modern package. The work is
% released under the GUST Font License. See the
% MANIFEST-Latin-Modern.txt and README-Latin-Modern.txt files
% for the details. For the most recent version of this license
% see http://www.gust.org.pl/fonts/licenses/GUST-FONT-LICENSE.txt
% or http://tug.org/fonts/licenses/GUST-FONT-LICENSE.txt
```

[10] Le prime 6 righe sono state leggermente riformattate per far rientrare il testo dentro la giustezza di questa guida.

```
\ProvidesFile{t1lmr.fd}[2009/10/30 v1.6 Font defs for Latin Modern]
\DeclareFontFamily{T1}{lmr}{}
\DeclareFontShape{T1}{lmr}{m}{n}%
     {<-5.5>    ec-lmr5     <5.5-6.5> ec-lmr6
      <6.5-7.5> ec-lmr7     <7.5-8.5> ec-lmr8
      <8.5-9.5> ec-lmr9     <9.5-11>  ec-lmr10
      <11-15>   ec-lmr12
      <15-> ec-lmr17
      }{}
\DeclareFontShape{T1}{lmr}{m}{sl}%
     {<-8.5>    ec-lmro8    <8.5-9.5> ec-lmro9
      <9.5-11>  ec-lmro10   <11-15>   ec-lmro12
      <15-> ec-lmro17
      }{}
\DeclareFontShape{T1}{lmr}{m}{it}%
     {<-7.5>    ec-lmri7
      <7.5-8.5> ec-lmri8    <8.5-9.5> ec-lmri9
      <9.5-11>  ec-lmri10   <11->   ec-lmri12
      }{}
\DeclareFontShape{T1}{lmr}{m}{sc}%
     {<-> ec-lmcsc10}{}
\DeclareFontShape{T1}{lmr}{m}{ui}%
     {<-> ec-lmu10}{}
%
% Is this the right 'shape'?:
\DeclareFontShape{T1}{lmr}{m}{scsl}%
     {<-> ec-lmcsco10}{}
%%%%%%% bold series
\DeclareFontShape{T1}{lmr}{b}{n}
     {<-> ec-lmb10}{}
\DeclareFontShape{T1}{lmr}{b}{sl}
     {<-> ec-lmbo10}{}
%%%%%%% bold extended series
\DeclareFontShape{T1}{lmr}{bx}{n}
     {<-5.5>   ec-lmbx5      <5.5-6.5> ec-lmbx6
      <6.5-7.5> ec-lmbx7      <7.5-8.5> ec-lmbx8
      <8.5-9.5> ec-lmbx9      <9.5-11>  ec-lmbx10
      <11->   ec-lmbx12
      }{}
\DeclareFontShape{T1}{lmr}{bx}{it}
     {<-> ec-lmbxi10}{}
\DeclareFontShape{T1}{lmr}{bx}{sl}
     {<-> ec-lmbxo10}{}
%%%%%%% Font/shape undefined, therefore substituted
\DeclareFontShape{T1}{lmr}{b}{it}
```

```
    {<->sub * lmr/b/sl}{}
\endinput
%%
%% End of file `t1lmr.fd'.
```

Dopo le prime righe di informazioni sulle licenze, il codice comincia con la identificazione del file, la sua data e la sua versione. Poi specifica che quanto segue riguarda la "FontFamily" identificata dalla codifica *T1* e dalla famiglia `lmr`. Poi definisce una dozzina di forme abbinate alle loro serie, specificandone i corpi disponibili. La prima dichiarazione definisce i font di serie media (`m`) e di forma normale (`n`), definisce, cioè, il tondo medio. Il quinto argomento elenca i corpi disponibili e i nomi dei file dei font con i quali rappresentare quei corpi. Ogni font reale (*ec-lmr5, ec-lmr6, . . . , ec-lmr17*) può generare corpi diversi contenuti in un intervallo: per esempio l'intervallo indicato con `<9.5-11>` indica tutti i corpi da 9,5 pt incluso a 11 pt escluso. I font più piccoli sono caratterizzati da un intervallo aperto `<-5.5>` che manca dell'estremo inferiore, quindi il font *ec-lmr5* serve per rappresentare i caratteri di corpo piccolissimo (idealmente fino a 'zero') fino al corpo 5,5 pt escluso. I font più grandi sono identificati da un altro intervallo aperto `<15->` per cui il font slshape ec-lmr17 serve per rappresentare tutti i corpi da 15 pt compreso fino a corpi molto grandi (idealmente infinitamente grandi). In questo momento il testo viene composto con il font *T1/lmr/m/n/10* che seleziona mediante la macro `\T1/lmr/m/n/10` il cui significato è `select font ec-lmr10`. Come si vede, questa macro strana, il cui nome contiene anche delle barre e delle cifre, è costruita con il nome della codifica, della famiglia, della serie, della forma e del corpo e serve proprio a selezionare il font *ec-lmr10*; sono tutte informazioni ricavate dal file di descrizione del font corrente.

Si noti la forma `scsl`: essa corrisponde al maiuscoletto inclinato, non direttamente accessibile con i comandi standard di LaTeX, almeno quelli per l'utente; questi per altro potrebbe usare i comandi di basso livello per ottenere IL MAIUSCOLETTO INCLINATO: il comando da usare è:

```
\fontshape{scsl}\selectfont
```

Un'altra forma non direttamente accessibile è la forma corsiva diritta, `ui`, che in questo testo è stato usato diverse volte: *corsivo diritto* con un comando definito apposta, ma sempre accessibile mediante i comandi di basso livello:

```
\fontshape{ui}\selectfont
```

Tra le serie c'è anche il nero non esteso disponibile almeno per le forme normale e inclinata (`sl`) si confronti infatti il **nero non esteso** con il **nero esteso** – in questo caso la differenza non è molto visibile, ma su un testo più lungo è decisamente percepibile. Il nero non esteso è accessibile con i comandi di basso livello:

```
\fontseries{b}\selectfont
```

L'ultima forma abbinata al nero non esteso riguarda il corsivo: ma la collezione non contiene nessun font reale che possa rappresentare il corsivo nero non esteso, quindi lo sostituisce con il font *lmr/b/sl*, cioè con il tondo inclinato nero non esteso. La sostituzione è indicata con la parola chiave `sub` che quando viene attivata, oltre ad eseguire la sostituzione, scrive anche un messaggio nel file `.log`. Esisterebbe anche la parola chiave `ssub` che esegue la sostituzione silenziosamente, cioè senza riportare nulla nel file `.log`.

Come si vede, dunque, quando si usano i comandi di alto o di basso livello per scegliere un font, il programma di composizione va a leggere[11] il file di descrizione per la codifica corrente (o quella specificata e per la famiglia corrente o quella specificata esplicitamente) e poi esso esamina ogni dichiarazione di forma fino a trovare il corpo giusto oppure la sostituzione predefinita.

Un utente che voglia caricare una collezione, o anche una sola famiglia di font, è bene che controlli se esistono già i file di descrizione; in mancanza dovrebbe crearseli se non vuole ridursi ad usare i comandi nativi, come si fa ancora oggi quando si usa Plain TeX.

Il lettore non si spaventi, se dovesse crearsi dei file di descrizione: quelli relativi ai font Latin Modern sono relativamente complessi perché dispone di famiglie molto ricche di forme e di serie; non solo, ma dispone anche di corpi ottici con cui cambiare leggermente il disegno dei glifi omonimi a seconda del corpo in modo da migliorarne la leggibilità per i corpi minori e non annerire troppo i corpi maggiori. Se ne è già discusso (vedi tabella 18.2). Tant'è che il file di descrizione per i font TX è molto più semplice:

```
\ProvidesFile{t1txr.fd}
        [2000/12/15 v3.1]

\DeclareFontFamily{T1}{txr}{}
\DeclareFontShape{T1}{txr}{m}{n}{% medium
     <->t1xr%
}{}
\DeclareFontShape{T1}{txr}{m}{sc}{% cap & small cap
     <->t1xsc%
}{}
\DeclareFontShape{T1}{txr}{m}{sl}{% slanted
     <->t1xsl%
}{}
\DeclareFontShape{T1}{txr}{m}{it}{% italic
     <->t1xi%
}{}
\DeclareFontShape{T1}{txr}{m}{ui}{% unslanted italic
     <->ssub * txr/m/it%
}{}
```

[11] In realtà questa lettura viene fatta una volta sola e il suo contenuto resta memorizzato all'interno del programma.

```
\DeclareFontShape{T1}{txr}{bx}{n}{% bold extended
     <->t1xb%
}{}
\DeclareFontShape{T1}{txr}{bx}{sc}{% bold extended cap & small cap
     <->t1xbsc%
}{}
\DeclareFontShape{T1}{txr}{bx}{sl}{% bold extended slanted
     <->t1xbsl%
}{}
\DeclareFontShape{T1}{txr}{bx}{it}{% bold extended italic
     <->t1xbi%
}{}
\DeclareFontShape{T1}{txr}{bx}{ui}{% bold extended unslanted italic
     <->ssub * txr/bx/it%
}{}
\DeclareFontShape{T1}{txr}{b}{n}{% bold
     <->ssub * txr/bx/n%
}{}
\DeclareFontShape{T1}{txr}{b}{sc}{% bold cap & small cap
     <->ssub * txr/bx/sc%
}{}
\DeclareFontShape{T1}{txr}{b}{sl}{% bold slanted
     <->ssub * txr/bx/sl%
}{}
\DeclareFontShape{T1}{txr}{b}{it}{% bold italic
     <->ssub * txr/bx/it%
}{}
\DeclareFontShape{T1}{txr}{b}{ui}{% bold unslanted italic
     <->ssub * txr/bx/ui%
}{}
\endinput
```

Si nota, infatti, che tutti gli intervalli dei corpi sono aperti da entrambi i lati, nel senso che qualunque corpo, da piccolissimo a grandissimo, viene ricavato rimpicciolendo o ingrandendo i segni messi a disposizione da un unico font reale. Inoltre sono presenti molte sostituzioni silenziose. In sostanza la disponibilità di forme e di serie con in font TX è molto minore di quella che si può avere con i font Latin Modern.

18.8.10 Ricapitolazione

Da quanto esposto si vede dunque che l'installazione di nuovi font estranei alle distribuzioni ufficiali di TEX Live e di MiKTEX non è una operazione che si risolva con quattro click di mouse. Richiede attenzione e precisione, ma non si tratta in generale di operazioni difficili: delicate sì, ma non difficili. Avendo capito quale sia il meccanismo delle mappe e dei font accessibili ai vari programmi di

composizione o di trasformazione, il tutto diventa più facile e la successione delle operazioni diventa del tutto naturale.

Ciò non toglie che non possano sorgere problemi con i sistemi operativi che vengono usati sulle diverse macchine, sistemi operativi che hanno l'abitudine di cambiare assai spesso, talvolta senza retrocompatibilità; chi scrive aggiunge nuovi font abbastanza spesso, certo non tutti i giorni ma almeno una volta all'anno, e trova che il meccanismo operativo oggigiorno sulla distribuzione TeX Live sia ottimo, e non incontra più i problemi che invece doveva affrontare negli anni passati.

Non è quindi il caso di spaventarsi; si possono aggiungere altri font ai moltissimi già distribuiti con le varie installazioni disponibili; per un solo font può essere una questione di un quarto d'ora; per una collezione di font le operazioni possono susseguirsi molto più rapidamente.

18.8.11 Riassunto delle operazioni di gestione dei font e della loro connessione con l'intero sistema TeX

Solo pdflatex

Vale la pena di riassumere graficamente il modo di operare del sistema TeX quando si usi latex o pdflatex. Molte delle informazioni sono sparse in diversi altri capitoli, ma qui, dove si parla di font, sembra il posto giusto per collegare tutto assieme.

Nelle figure 18.9–18.13 i simboli grafici hanno i significati seguenti:

Font I file che si riferiscono ai font in senso stretto sono rappresentati da cerchi gialli.

Programmi di sistema Sono rappresentati da rettangoli rosso chiaro.

Programmi esterni Sono rappresentati da rettangoli azzurri.

File accessori per la gestione dei font Sono rappresentati da cerchi verdi.

File finali Sono rappresentati da cerchi marroncino chiaro; 'file finale' indica il file che contiene un documento già composto.

Risorse esterne Sono gruppi di file di programmi e/o di font da reperire gratuitamente in rete, o onerosamente presso i rivenditori specifici. Sono rappresentati da ellissi blu chiaro.

Uscite L'uscita su schermo, su carta, su lastre per fotolitotipografia, o su altri supporti sono rappresentati mediante ellissi arancione.

Nella figura 18.9 sono rappresentati due dei processi di composizione descritti in questo testo, quello attraverso latex e quello attraverso pdflatex. Si noti la semplicità del processo attraverso latex, a confronto con la relativa complicazione attraverso il processo pdflatex[12]. Il primo richiede solo il file sorgente e i file metrici dei font usati, e in uscita presenta il risultato 'finale' mediante un file `.dvi`. Questo file è finale solo in senso logico, perché, comunque, per essere utilizzato ha ancora bisogno di trasformazioni, specialmente se vi sono state incluse delle immagini ottenute da file esterni.

[12] Questo schema vale anche per XeLaTeXe LuaLaTeX; sono le figure successive che valgono solo per latex o pdflatex.

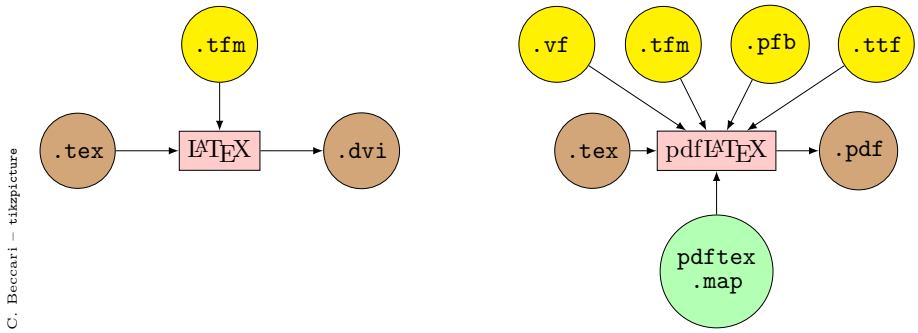


Figura 18.9: Schema grafico del processo di composizione

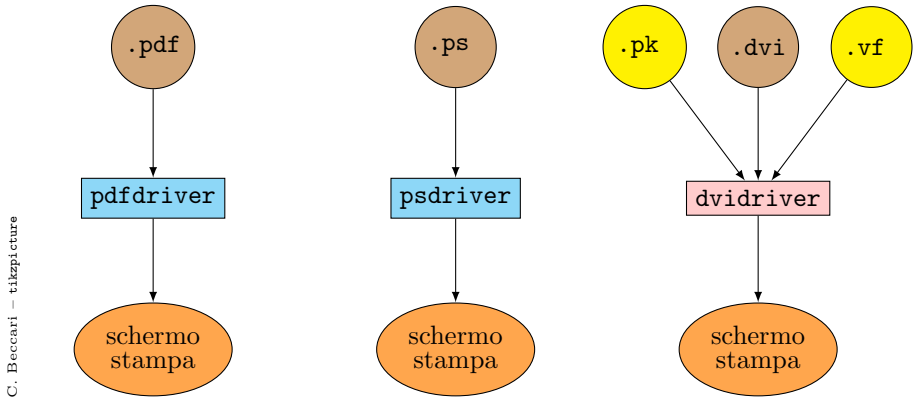


Figura 18.10: Presentazione all'esterno dei risultati della composizione

Al contrario il processo di composizione mediante pdflatex è completo e il file finale si può considerare definitivo; per fare ciò, però, esso ha bisogno di tutti i file concernenti i font, specialmente quelli di tipo vettoriale.

Il file `.dvi` può essere visualizzato e/o stampato direttamente mediante software che solitamente fa già parte del corredo del sistema TeX; sarà il visualizzatore YAP di MiKTeX, oppure xdvi di Linux oppure altri programmi simili. Lo stesso può dirsi del file `.pdf` in uscita da pdflatex, così come del file `.ps` in uscita dal convertitore dvips, nel caso che sia richiesto questo formato per l'uso successivo. Nella figura 18.10 sono rappresentati graficamente i processi di presentazione mediante i vari programmi specifici e sono messi in evidenza i file di cui necessitano, con particolare rilievo per quelli che riguardano i font.

Generalmente però il file `.dvi` ha bisogno di essere trasformato o in file `.pdf` o in file `.ps`; quest'ultimo a sua volta può richiedere la trasformazione ulteriore in file `.pdf`; il processo di conversione è rappresentato graficamente nella figura 18.11, ancora evidenziando l'importanza dei file concernenti i font.

Come si vede da queste figure, i file direttamente o indirettamente legati

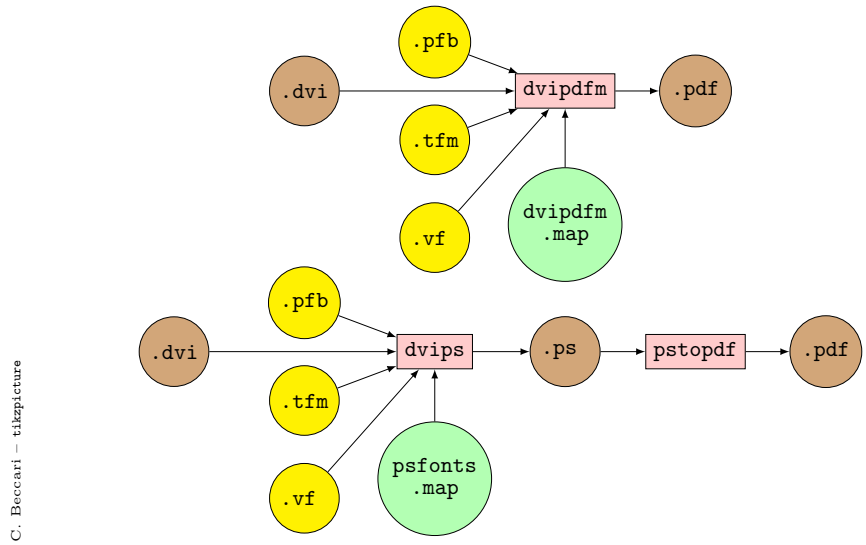


Figura 18.11: Processi di conversione dei vari formati di uscita dai programmi di composizione

ai font sono tanti e spesso bisogna crearli apposta con il supporto delle utility fornite con il sistema TeX o con l'ausilio di programmi esterni. La figura 18.12 mostra in forma schematica le risorse esterne da cui trarre i font necessari e come trasformare questi ultimi in modo da renderli usabili con il sistema TeX. Si vede chiaramente che i font in formato METAFONT sono i più facili da usare, il che è logico, visto che TeX e METAFONT sono programmi 'fratelli' entrambi concepiti da Donald Knuth e sono sempre distribuiti assieme e completi di una buona collezione di file sorgente di tipo `.mf`. Ma i font vettoriali distribuiti secondo lo standard Adobe (PostScript Type 1), oppure i font TrueType, nati da un progetto comune della Adobe e della Microsoft, richiedono ulteriori trattamenti, se non altro perché non sono dotati dei file `.tfm`, che bisogna quindi generare; spesso hanno anche codifiche diverse, certamente diverse da quelle che TeX si aspetta, e quindi sono necessarie conversioni a diversi livelli. I programmi che operano queste conversioni e il loro modo di procedere sono illustrati schematicamente nella figura 18.12; in questa figura è indicata anche la conversione dei font `.mf` in font di tipo vettoriale (mediante il programma mftrace); i font vettoriali sono utilissimi, specialmente per la visualizzazione su schermo dove è necessario poter ingrandire o rimpicciolire la schermata, e quindi il suo contenuto, secondo le necessità di lettura.

In particolare le mappe parziali dei font disponibili sulla macchina specifica sulla quale viene eseguita la composizione vanno unite in modo coerente fino a formare le mappe generali di cui necessitano i programmi di conversione dei file finali. Il procedimento è indicato schematicamente nella figura 18.13; il programma indicato con il nome 'editor' è un qualunque programma di gestione

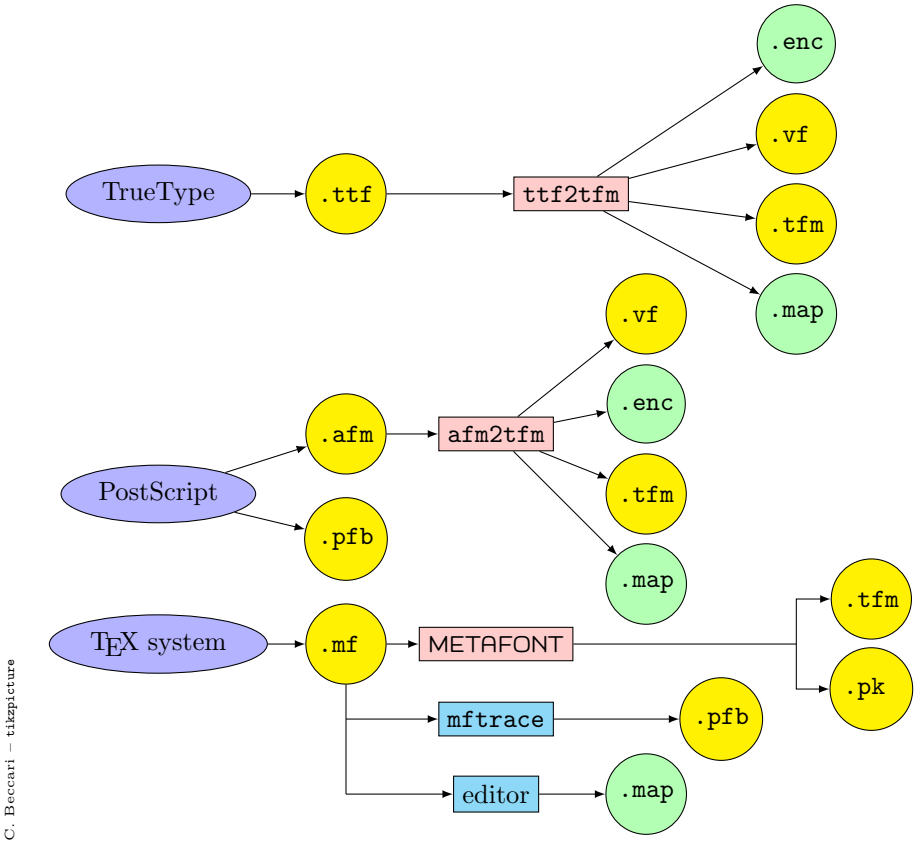


Figura 18.12: Trasformazioni di formato dei font e generazione dei file ausiliari

di file testuali ASCII; potrà essere Notepad dei sistemi Windows, oppure vim dei sistemi Linux, ma va benissimo anche lo *shell editor* usato per gestire i file sorgente da comporre con il sistema TEX; il file `updmap.cfg` in uscita dall'editor è quello che crea l'utente per i suoi font privati; gli altri file con lo stesso nome sono quelli del sistema TEX e non vanno toccati assolutamente dall'utente.

18.9 Conclusioni

Tutte queste operazioni, descritte forse con troppo dettaglio, possono impaurire il neofita; tuttavia ci si può consolare pensando che l'installazione di un nuovo font non è una operazione da eseguirsi tre volte al giorno; d'altra parte se così fosse, queste operazioni finirebbero per venire eseguite in modo del tutto automatico e non apparirebbero così intricate.

Ci si potrebbe domandare perché il sistema TEX, che è dotato di un gran numero di font accessori, non sia dotato anche di un programma che consenta di

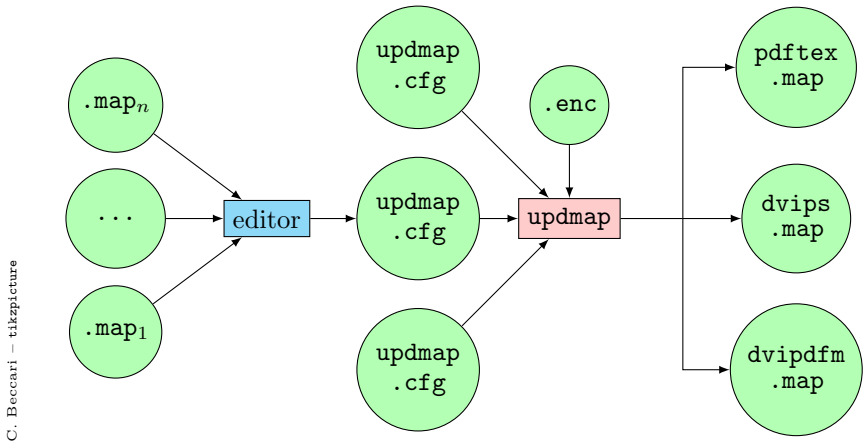


Figura 18.13: Gestione delle mappe dei font

eseguire tutte le operazioni indicate in questi ultimi paragrafi in modo automatico. Buona domanda. Una risposta possibile è che per ogni font o collezione di font bisogna agire in modo diverso da ogni altro font o collezione. I nomi, le codifiche, il contenuto dei file che contengono le informazioni che descrivono i vari font, potrebbero essere forse oggetto di un programma o di uno script in linguaggio bash, oppure python, oppure perl, oppure java, oppure ..., ma le possibilità di errore sono così numerose che il lavoro del programmatore diventerebbe troppo oneroso, specialmente in relazione al fatto che l'installazione di nuovi font non avviene, come si è detto, tre volte al giorno.

Chi scrive generalmente reinstallava i propri font non standard una volta all'anno, quando aggiornava il suo sistema; talvolta aggiungendo un nuovo font, di cui creava a mano la mappa e di cui aggiungeva il nome ad un file personale `local.map`, già elencato nel file di configurazione locale `updmap.map`, e tutto il resto si ripeteva di anno in anno senza bisogno di avere un comportamento diverso per ogni nuovo font.

Oggi le cose sono molto più semplici. Fermo restando che il sistema TeX non può sapere se l'utente ha creato nuovi font, l'utente ha ancora il dovere di elencare i suoi nuovi font nella sua mappa personale. Ma fatto questo ogni nuova installazione annuale del sistema TeX scopre da sola l'esistenza di diversi file di configurazione e li concatena in un unica configurazione per usarla onde aggiornare la mappa generale dei font.

Ad ogni aggiornamento dell'installazione (che anche altrove si è raccomandato di eseguire ogni 7-10 giorni, massimo due settimane) ogni volta che il sistema si arricchisce di nuovi font rigenera la mappa, eseguendo sempre la concatenazione dei vari file di configurazione, per cui la manutenzione delle mappe è diventata una cosa di nessun peso.

Capitolo 19

LaTeX: nuovi comandi

19.1 Introduzione

Uno dei punti di forza del linguaggio di programmazione che sostiene il sistema TeX è costituito dal fatto di poter definire nuovi comandi, nuove istruzioni.

Gli interpreti tex, o pdftex, o xetex, o luatex che interpretano i comandi dei vari tipi di mark-up dei file sorgente sono in grado di eseguire alcune centinaia di comandi base, comandi 'nativi' o 'primitivi'; primitivi non nel senso che siano rozzi, ma che sono quelli a cui tutti gli altri si riferiscono; i comandi derivati sono per lo più definiti nei vari file di formato; `plain.tex` per il mark-up di Plain TeX; `latex.ltx` per il mark-up di LaTeX, eccetera.

Questi file di formato *definiscono* dei nuovi comandi, detti *macro*, attraverso i comandi primitivi o attraverso quelli derivati e già definiti.

Quando l'interprete incontra una macro, la deve sviluppare, cioè deve ricercarne in memoria la traduzione in base alla definizione specifica di quella macro; se questa definizione contiene le azioni da eseguire specificate solamente mediante comandi primitivi, l'interprete li esegue, altrimenti sequenzialmente ricerca i comandi derivati presenti nella definizione, ripetendo il processo finché lo sviluppo contiene solo comandi primitivi che finalmente esegue.

Ogni macro può essere definita in modo che possa lavorare su *argomenti*, cioè parti variabili che cambiano di volta in volta che si invoca la macro. Gli argomenti possono essere *delimitati* o *non delimitati*. All'interno dei formati si fa un grande uso di argomenti delimitati, ma i comandi per gli utenti, specialmente con LaTeX, fanno un uso minimo di argomenti delimitati. Vedi però il capitolo 30 dove si parla dei recenti aggiornamenti e dell'introduzione delle funzionalità del pacchetto *xparse* nel nucleo di LaTeX; grazie a queste funzionalità, a cui ci si riferirà con alcuni esempi in questo capitolo, si possono definire comandi che accettano argomenti variamente delimitati con i quali si possono fare cose inimmaginabili prima di quegli aggiornamenti.

Gli argomenti non delimitati sono costituiti da un solo *token*, per esempio una sola lettera o una sola cifra, e non richiedono di venire racchiusi dentro nessuna struttura che ne limiti la lunghezza. Altrimenti devono essere racchiusi fra parentesi graffe. Nei capitoli precedenti si sono visti quasi sempre argomenti non delimitati indicati nelle istruzioni sintattiche con `{`⟨*argomento*⟩`}`. LaTeX usa raramente gli argomenti delimitati, ma quando lo fa, li delimita solo mediante parentesi quadre; è quanto si è indicato con `[`⟨*argomento*⟩`]`; nell'ambiente *picture* sono delimitate le coordinate, racchiuse fra parentesi tonde e indicate nelle costruzioni sintattiche con `(`⟨x⟩`,`⟨y⟩`)`. Ma per il resto è difficile trovare altri comandi destinati al compositore che facciano uso di argomenti delimitati.

Qui si illustreranno succintamente le definizioni di nuovi comandi, le ridefinizioni di comandi già esistenti; le definizioni di nuovi ambienti e le ridefinizioni di ambienti già esistenti.

Come preannunciato sopra, grazie al nuovo linguaggio L3, LaTeX 3, sono disponibili file di estensione, come *xparse* (documentazione `texdoc xparse`), che permettono agli utenti di definire facilmente i comandi con argomenti delimitati e con diversi tipi di argomenti facoltativi. Merita approfondire questo argomento leggendo la documentazione del pacchetto citato. Si noti che dal 2020 le funzionalità di questo pacchetto sono integrate nel nucleo di LaTeX, per cui esse sono disponibili senza bisogno di caricare
packxparse.

Ugualmente importante è il pacchetto *xfp* che permette di fare calcoli estremamente precisi in virgola mobile e notazione scientifica usando stringhe di numeri decimali; anche le funzionalità di questo pacchetto sono già integrate nel nucleo di LaTeX, e non è necessario caricarlo; resta disponibile la documentazione leggibile con `texdoc xfp`.

19.2 Le definizioni di comandi nuovi con LaTeX 2ε

Per definire un nuovo comando si usa la seguente sintassi:

```
\newcommand{⟨macro⟩}[⟨num-argomenti⟩][⟨default⟩]{⟨definizione⟩}
\newcommand*{⟨macro⟩}[⟨num-argomenti⟩][⟨default⟩]{⟨definizione⟩}
\providecommand{⟨macro⟩}[⟨num-argomenti⟩][⟨default⟩]{⟨definizione⟩}
\providecommand*{⟨macro⟩}[⟨num-argomenti⟩][⟨default⟩]{⟨definizione⟩}
\DeclareRobustCommand{⟨macro⟩}[⟨num-argomenti⟩][⟨default⟩]{⟨definizione⟩}
\DeclareRobustCommand*{⟨macro⟩}[⟨num-argomenti⟩][⟨default⟩] {⟨definizione⟩}
```

dove ⟨*num-argomenti*⟩ è il numero degli argomenti su cui opera la ⟨*macro*⟩ dei quali il primo può essere facoltativo; esso è facoltativo se viene espressa anche la seconda opzione incluse le sue parentesi quadre; se nell'invocazione della macro che accetta un argomento facoltativo non lo si specifica, esso riceve il valore di ⟨*default*⟩. Nella definizione della ⟨*macro*⟩ gli argomenti, al massimo in numero di 9, sono indicati con il loro numero progressivo `#1`, `#2`, eccetera. La ⟨*macro*⟩, cioè il suo nome, può essere costituita solo:

- dal segno di backslash \ seguito da una stringa formata solamente da lettere dell'alfabeto latino maiuscole o minuscole, ricordando che LaTeX distingue le une dalle altre; oppure
- dal segno di backslash \ seguito da un solo segno non alfabetico; oppure
- da un carattere attivo; un carattere attivo è un segno che ha ricevuto in precedenza l'etichetta di carattere attivo; LaTeX dichiara carattere attivo il segno ~ al quale ha dato il significato di *spazio non separabile* (in corrispondenza del quale non può cadere la fine di una riga); `babel` con l'opzione di quasi tutte le lingue che usano caratteri accentati, fra cui l'italiano, definisce attivo anche il carattere ".

Il comando `\newcommand`, con o senza asterisco, controlla se la ⟨*macro*⟩ sia già stata definita e, nel caso, emette un messaggio d'errore. Il nuovo comando potrebbe risultare fragile (la fragilità di comandi è un concetto delicato, descritto nel paragrafo 19.6 e descritto anche nel capitolo 29), mentre `\DeclareRobustCommand` crea una definizione robusta, cioè "non fragile"; vedi il paragrafo 19.6. I comandi senza asterisco creano definizioni i cui argomenti possono essere formati da più capoversi, mentre i comandi con l'asterisco creano comandi i cui argomenti possono essere formati anche da diverse parole, ma tali da non costituire un capoverso; nel caso un argomento contenga esplicitamente o implicitamente un comando di "fine capoverso", l'esecuzione del comando con un simile argomento genera un errore.Infine i comandi `\providecommand`, con o senza asterisco definiscono una macro solo se essa non è mai stata definita prima.

Si raccomanda agli utenti di non "giocherellare" con i caratteri attivi; qui li si è nominati perché è giusto sapere che esistono e che cosa sono, ma è meglio lasciarli stare. È meglio lasciare stare anche i comandi che usino solo o anche caratteri non alfabetici, come per esempio il carattere `@`.[1]

19.2.1 Definizione del logo di MacTeX

Come primo esempio si riporta quanto è stato definito nel preambolo di questo documento, la definizione del comando per scrivere il logo di MacTeX (vedi i comandi robusti nel paragrafo 19.6):

```
\DeclareRobustCommand*\MacTeX{Mac\-\TeX}
```

Come si vede non ci sono argomenti né obbligatori né facoltativi per la definizione della macro `\MacTeX`; la ⟨*definizione*⟩ si avvale di altri comandi: `\-` per consentire una cesura dopo la prima parte del logo, e `\TeX` per comporre la seconda parte formata dal logo di TeX.

[1] Nei file di formato compaiono una quantità di definizioni di comandi che contengono il carattere `@` come se fosse una lettera; all'interno del formato questo carattere è definito come lettera alfabetica, ma fuori del formato, quando il compositore usa LaTeX, esso è ripristinato alla categoria di 'non-lettera'; la cosa è fatta di proposito affinché i compositori non si mettano a giocare con i caratteri non alfabetici.

19.2.2 Definizione del comando \cs

Come secondo esempio si porta la definizione del comando `\cs` che è servito tante volte per scrivere i nomi dei comandi con carattere a spaziatura fissa e preceduti dal segno di backslash senza che LaTeX li eseguisse, ma li considerasse solo delle stringhe letterali:

```
\newcommand*{\cs}[1]{\texttt{\char92#1}}
```

Il comando accetta un solo argomento obbligatorio e la sua sintassi diventa perciò

`\cs{`⟨*nome della macro senza backslash*⟩`}`

La definizione della macro[2], oltre a contenere il comando `\texttt` per comporre il suo argomento con caratteri a spaziatura fissa, contiene anche il comando primitivo `\char` che si aspetta un numero decimale, oppure esadecimale, oppure ottale; dalle tabelle dei caratteri si vede che è più semplice riferirsi all'indirizzo decimale, invece che a quelli ottale o esadecimale, e si vede che il segno di backslash è nella casella con indirizzo 92; usando il comando `\char` e l'indirizzo nella tabella dei caratteri, il segno identificato viene trascritto tale e quale, prescindendo dal significato che esso potrebbe avere per LaTeX, cioè quello di iniziatore di nomi di macro: `\cs` è diverso da `\char92cs` il primo è un comando, il secondo è il nome di un comando.[3]

19.2.3 Il comando \\

Nel preambolo di questo documento non sono stati definiti comandi con argomento facoltativo; si prenderà un esempio da altri documenti; prendiamo una possibile definizione del comando `\\`:

```
\newcommand*{\\}[1][0pt]{\newline\vspace{#1}}
```

Questa non è la definizione completa che compare all'interno del file di formato, ma rende l'idea. Il comando `\\` accetta un argomento; siccome ne è specificato il valore di default, questo primo ed unico argomento è facoltativo; se lo si vuole indicare lo si deve scrivere fra parentesi quadre. Se non lo si indica affatto, viene usato il suo valore di default che viene passato tale e quale alla macro

[2] In realtà la definizione indicata è solo una versione preliminare; quando si è deciso di comporre anche l'indice analitico, si è cambiata la definizione includendovi il comando `\index` corredato di un opportuno argomento; la necessità di usare il segno `@`, che ha un significato particolare per makeindex, ci ha obbligati ad una definizione decisamente più complessa, la cui spiegazione esula da questo testo. Tuttavia è importante notare come la modifica della sola definizione di `\cs` ha permesso di inviare all'indice analitico tutti i comandi descritti, senza bisogno di inserire il comando `\index` ogni volta che li si citava.

[3] Invece di `\char92` si sarebbe potuto scrivere `\textbackslash`; ma questo comando agisce solo in modo testo, mentre `\char92` agisce anche in modo matematico. È vero che non lo si è mai usato in modo matematico, ma non si sa mai...

`\vspace` che esegue una spaziatura verticale pari all'ammontare specificato con l'argomento esplicito o di default; la spaziatura verticale di default è di 0 pt, cioè è una spaziatura nulla.

19.2.4 i comandi di definizione asteriscati

Il lettore attento avrà notato che in due dei tre esempi un asterisco segue il comando `\newcommand`; questo asterisco facoltativo esprime una variante del comando che è consigliabile usare ogni volta che il comando debba ricevere degli argomenti; come già detto, ma lo si evidenzia di nuovo, senza asterisco il comando può ricevere argomenti arbitrariamente lunghi, fatti anche di numerosi capoversi; con l'asterisco il comando può ricevere argomenti formati solo da una stringa di caratteri che non contenga né esplicitamente né implicitamente una terminazione di capoverso; questa come si ricorda, è costituita da una riga del file sorgente completamente vuota oppure dal comando esplicito `\par`.

Quando si definiscono comandi che accettano solo una o poche parole, certamente non più di un capoverso, è prudente inserire l'asterisco, perché nel caso che ci si dimentichi di scrivere la graffa di chiusura, viene emesso subito un messaggio d'errore che consente al compositore di intervenire per correggerlo. È bene ricordare che l'omissione di una graffa chiusa rappresenta l'errore più frequente quando si compone il testo sorgente.

19.2.5 I capilettera

Nel capitolo 20 si parlerà dei *capilettera*; vi si descriverà il pacchetto *lettrine* che consente di inserire queste grandi lettere maiuscole all'inizio del primo capoverso di un capitolo o di un paragrafo. Nell'esemplificare l'uso dei capilettera in quel capitolo *non* si è fatto uso delle macro disponibili con il pacchetto *lettrine* ma si è usato il comando `\capolettera` descritto qui di seguito insieme alle motivazioni per non aver usato il pacchetto citato.

La macro `\capolettera` è definita nel file `MacroGuida.sty` che raccoglie le macro usate per comporre questo testo. Questa macro `\capolettera` non ricorre al pacchetto *lettrine* perché sul forum di GUIT è stato notato che *lettrine* funziona bene solo con i font vettoriali che dispongono solamente di un corpo da ingrandire e rimpicciolire a seconda delle esigenze. Qui si usano in prima istanza i font Latin Modern (scalabili con continuità a tratti) o, in seconda istanza, i font Computer Modern[4], che sono dotati di tutti i corpi necessari. Si è preferito quindi definire una macro apposita che svolgesse più o meno le stesse funzioni offerte dal pacchetto *lettrine*, ma non facesse riferimento implicito ad un solo corpo. La macro qui descritta produce un capolettera grande come due righe di testo e costruito in modo da essere a filo con la linea di base della seconda riga, ma in modo che la sua altezza non superi quella delle lettere in maiuscoletto che

[4] I font Computer Modern non sono scalabili, ma sono disponibili solo in certi corpi; però, usando il pacchetto *type1ec*, *senza* specificargli l'opzione *10pt*, i font CM-Super sono scalabili con continuità a tratti.

la seguono. Il suo scopo, infatti è quella di cominciare un capoverso nel modo seguente.

UN GRANDE CAPOLETTERA deve essere molto visibile come inizio di un capoverso ma richiede una transizione di forma dei caratteri che lo seguono per rendere meno brusco il passaggio dalle sue grandi dimensioni al corpo normale del font usato nel resto del capoverso. Per questo motivo la tradizione tipografica generalmente ricorre a qualche parola iniziale del capoverso, dopo il capolettera, composta in maiuscoletto.

La macro perciò richiede di specificare come primo argomento la lettera da ingrandire e, come secondo argomento, il completamento della prima parola seguito da qualche altra parola iniziale del capoverso:

```
\newlength\CLett              % Nuova dimensione

\newcommand*\capolettera[2]{% #1: lettera da ingrandire
                            % #2 testo da comporre in maiuscoletto
\setbox\z@\hbox{\scalebox{1.25}{\huge#1}}\CLett=\wd\z@%
\par\noindent
\hangindent\CLett\hangafter-2\relax
\raisebox{-\baselineskip}[\z@]\z@]{%
          \llap{\box\z@\kern1pt}}\textsc{#2}}
```

La definizione ricorre a comandi di base del linguaggio TeX, mescolati a comandi di LaTeX e del pacchetto *graphicx*. Per i dettagli si può guardare nel capitolo 29. Il capolettera si ottiene mettendo nella scatola numerata '0' (`\z@`) la lettera da ingrandire ma che viene inizialmente ottenuta con il comando `\huge` e poi viene ulteriormente ingrandita del 25%, per tenere conto che si tratta di una maiuscola (generalmente) senza discendenti.

Se ne prende la misura della larghezza per far rientrare le righe della giusta quantità; ma poi sono `\hangindent` e `\hangafter` che creano lo spazio nelle prime due righe per mettere la lettera ingrandita; `\hangindent` riceve come argomento l'ammontare del rientro e `\hangafter` riceve il numero di righe per le quali vale il rientro; il segno si riferisce al margine destro (segno più) o sinistro (segno meno) per il quale si esegue l'operazione del rientro. Alla fine del capoverso questi due parametri 'primitivi' di TeX vengono ripristinati a valori nulli, cosicché queste impostazioni valgono solo per il capoverso corrente.

Per collocare la lettera si usa il comando LaTeX `\raisebox` che mette il suo argomento in una scatola, e la solleva della quantità specificata nel primo parametro (se essa è negativa la abbassa) e tratta la scatola come se avesse altezza e profondità pari ai due ulteriori parametri facoltativi (in questo caso entrambi nulli, cosicché non si tiene conto dell'altezza e, specialmente, della profondità della lettera ingrandita). Questa lettera viene inserita dentro una scatola di larghezza nulla `\llap`, ma con il contenuto sporgente a sinistra; vi viene posto anche un poco di spazio (1 pt) per scostare un minimo la lettera ingrandita dalle rientranze delle due righe. Il secondo argomento viene poi inserito tutto composto in maiuscoletto.

Il pacchetto *lettrine* consente di corredare il comando del capolettera con diverse opzioni, cosicché lo si possa fare sporgere in alto di più o di meno, in modo che occupi lo spazio di un numero di righe a scelta, e altre finezze, ma sembra, appunto, che funzioni correttamente solo con i font dotati di un solo carattere di corpo fisso, ma scalabile a piacere.

19.2.6 Un titolo di sezione da trattare in modo speciale

Un altro esempio istruttivo si presenta quando è necessario usare un comando di sezionamento che inserisca qualcosa di diverso dal solito sia nel suo titolo, sia in quello che viene elencato nell'indice, sia in quello che appare nelle testatine. I comandi normali del nucleo di LaTeX dispongono di un argomento facoltativo indicato con la dicitura ⟨*titolo breve*⟩:

\⟨*comando*⟩[⟨*titolo breve*⟩]{⟨*titolo lungo*⟩}

dove \⟨*comando*⟩ può essere `\part`, `\chapter`, `\section`, eccetera. Il ⟨*titolo breve*⟩ appare sia nella testatina sia nell'indice generale, ma non è possibile con questi comandi mettere due ⟨*titoli brevi*⟩ uno nell'indice e l'altro, diverso, nella testatina.

Non è il solo, ma uno dei motivi per fare questo è che se si usa il pacchetto *hyperref* i segnalibri ricevono lo stesso testo che va nell'indice, ma *hyperref* non è capace di gestire nei segnalibri alcuni comandi fragili e/o alcuni comandi che cambiano la famiglia, serie o forma dei caratteri. Questo è successo in questo testo, per esempio, nella composizione del paragrafo 11.5; dove il nome del pacchetto *biblatex* doveva apparire in minuscolo nella testatina, e doveva apparire in tondo minuscolo nei segnalibri, mentre nell'indice poteva apparire nello stile dei nomi dei pacchetti. Impossibile ottenere questi risultati con le macro predefinite.

Tuttavia è possibile farlo a mano, e se lo si fa a mano con una sequenza di operazioni collaudate, questa serie di operazioni possono servire per costituire la definizione di una nuova macro. Si è accennato sopra che sarebbe possibile usare i potentissimi comandi di definizione del pacchetto *xparse*; qui si espone la strategia con un comando che si riferisce al comando `\section`, ma, *mutatis mutandis*, la stessa strategia si può applicare anche agli altri comandi.

Si è deciso di usare il comando `\section*`, che scrive solo il titolo non numerato del paragrafo e non manda nulla né all'indice, né alla testatina. Assieme a questo comando, si usano i comandi di basso livello per inviare materiale diverso all'indice e alle testatine delle pagine destre occupate dall'intero paragrafo.

Si è trovata definita la macro seguente:

```
1 \newcommand\Section[3]{%
2 % #1 := parte fissa del titolo della sezione
3 % #2 := comando particolare da applicare alla terza parte
4 % #3 := parte da trattare in modo particolare
5 \refstepcounter{section}%
```

```
6 \section*{\thesection\quad#1 #2{#3}}%
7   \addcontentsline{toc}{section}{%
8     \protect\numberline{\thesection}{#1
9     \ifdefined\texorpdfsring\expandafter\@firstoftwo\else
10    \expandafter\@secondoftwo\fi{\texorpdfstring{#2}{}}{}{#3}}}%
11  \markright{\MakeUppercase{#1} #2{#3}}%
12 }
```

In questa macro i tre argomenti hanno i significati specificati nelle righe di commento; ma con questa macro, il titolo del paragrafo 11.5 è stato composto semplicemente con

```
\Section{Il pacchetto}{\packstyle}{biblatex}
```

È necessaria qualche spiegazione.

1. Nella riga 1 si specifica che la macro deve ricevere tre argomenti obbligatori, i cui significati sono spiegati nelle righe 2, 3 e 4.
2. Nella riga 5 si incrementa di una unità il contatore delle sezioni; infatti, benché dopo si usi il comando `\section*` per gestire il titolo della sezione, si desidera che la sezione sia numerata e che si possa fare riferimento al suo numero mediante il solito meccanismo dei riferimenti incrociati che usano `\label` e `\ref`. Il comando `\refstepcounter` incrementa di una unità il contenuto del contatore `section` e lo rende accessibile al meccanismo dei riferimenti incrociati.
3. Il comando `\section*` non numera il paragrafo di cui scrive il titolo; ma se si vuole una sezione numerata bisogna provvedere esplicitamente ad inserire il numero prima del titolo alla giusta distanza; questa, nel nostro caso, è un quadrato (`\quad`).
4. Nelle righe 7—10 si provvede a inviare all'indice quanto vi deve venire scritto; per questo scopo si usa il comando `\addcontentsline` la cui sintassi è ricordata più avanti nel capitolo 29. Il suo primo argomento specifica con la sigla `toc` il file nel quale scrivere le informazioni; il secondo argomento specifica il livello logico di quello che vi si vuole inserire, ma il terzo argomento contiene un comando `\protect` che protegge da uno sviluppo intempestivo quanto segue, in particolare il comando `\numberline` con i suoi argomenti; questo comando scrive una voce nell'indice che comincia con un numero, prosegue con un titolo che, con una fila di puntini, guida l'occhio verso il numero della pagina appoggiato al margine destro dello specchio di stampa.
5. All'interno del terzo argomento di `\numberline` c'è un costrutto insolito, perché si esegue un test mediante `\ifdefined` per verificare che il comando `\texorpdfstring` sia definito; se lo è lo usa, se non lo è non lo usa. Sembra lapalissiano, ma non lo è. Infatti il test eseguito con `\ifdefined` segue la stessa sintassi di tutti i comandi condizionali nativi di TeX. Questi hanno il difettuccio di restare a mezz'aria a seconda di cosa contengano i

due rami del test; il primo, il test è vero, finisce con `\else` e il secondo, il test è falso, finisce con `\fi`. Per evitare questa apparente "debolezza" di questi comandi condizionali, i rami vero e falso vengono sostituiti da `\@firstoftwo` e `\@secondoftwo`, ma con `\expandafter` viene invertito l'ordine di esecuzione dei comandi; cioè nel ramo vero, benché `\else` sia scritto dopo `\@firstoftwo`, esso viene eseguito prima; la stessa cosa avviene nel caso falso, dove `\fi` viene eseguito prima di `\@secondoftwo`. Con questo artificio, usato a piene mani all'interno di ogni pacchetto, classe, file di formato, eccetera, del sistema TeX, si è sicuri che l'operazione condizionale non resti a mezz'aria. Successivamente ogni "frase" condizionale `\if...\else...\fi` deve essere seguita da due gruppi, dei quali il primo contenga i comandi da eseguire nel caso vero e il secondo quelli da eseguire nel caso falso. Le due macro interne `\@firstoftwo` e `\@secondoftwo` Servono appunto per leggere i due gruppi per restituire inalterato solo il primo o solo il secondo.

6. Nella macro in esame il comando `\texorpdfstring` è fornito dal pacchetto *hyperref*; esso accetta due argomenti: il primo serve per l'indice vero e proprio, mentre il secondo serve per i segnalibri; in questo caso esso scrive nell'indice l'argomento `#2` (cioè, nell'applicazione specifica, il comando `\packstyle`), ma non scrive niente nel file ausiliario che raccoglie le informazioni per i segnalibri; il terzo argomento racchiuso fra graffe, funziona da argomento solo per il comando contenuto nell'argomento `#2` (cioè `\packstyle`), ma questo terzo argomento viene scritto senza modificazioni e senza graffe nel file dei segnalibri. Il comando `\texorpdfstring` è utilissimo, ma gli utenti spesso se ne dimenticano, e vanno incontro ad errori di compilazione talvolta difficili da correggere.
7. Visto che in questo testo il pacchetto *hyperref* è stato caricato, il comando `\texorpdfstring` è sicuramente definito, quindi tutto quel codice per verificarne l'esistenza e provvedere in merito sembra inutile; lo si è messo comunque, perché la macro potrebbe essere recuperata per la composizione di altri documenti che ne avessero bisogno, ma non è detto quegli altri documenti facciano uso di collegamenti ipertestuali e che quindi carichino il pacchetto *hyperref*.
8. Finalmente nella riga 11 si arriva al testo per le testatine; `\markright` crea il materiale da inserire nella testatina della pagina di destra, che normalmente contiene il titolo del paragrafo corrente alla fine della pagina. Nello stile delle pagine predefinito nella classe *book*, usata per comporre questo testo, le testatine sono composte in tondo inclinato maiuscolo; ma qui non si può scrivere il nome del pacchetto *biblatex* in maiuscolo inclinato, ma si deve rendere in maiuscolo solo la prima parte della testatina; lo stile standard delle pagine provvede all'inclinazione dei caratteri.

19.2.7 Scalare il corpo

Componendo un documento con font scalabili con continuità eventualmente a tratti, si può ricorrere al comando `\setfontsize` descritto in questa sezione; Per cambiamenti di corpo globali, validi per l'intero documento si potrebbe usare il pacchetto*fontsize*, alla cui documentazione si rimanda il lettore; ma talvolta è utile disporre di un comando semplicissimo che permetta di scalare il corpo del font in uso senza ricorrere alle dichiarazioni di corpo predefinite; ovviamente, come per quelle dichiarazioni, il comando va usato dentro un gruppo affinché il suo effetto sia locale. In questo testo lo si è usato relativamente spesso per poter ridurre la lunghezza delle righe di testo all'interno di ambienti contenenti testi speciali, come la sintassi dei comandi o negli ambienti verbatim contenenti qualche riga un poco più lunga della giustezza.

La sintassi del comando è la seguente:

`\setfontsize{`⟨*corpo*⟩`}[`⟨*fattore*⟩`]`

dove il ⟨*corpo*⟩ è il corpo desiderato e il ⟨*fattore*⟩ è il coefficiente che determina lo scartamento a partire dal ⟨*corpo*⟩; questo fattore ha un valore di default pari a 1.2, che corrisponde al fattore di (quasi) tutti i font usati dal sistema TeX. Per definire questo comando è comodo usare non solo la sintassi specificata dalle funzionalità del pacchetto *xparse* già integrate nel nucleo di LaTeX, ma anche alle funzionalità del pacchetto *xfp* che però va caricato espressamente nel preambolo; questo pacchetto serve per eseguire calcoli in virgola mobile con una grande precisione; molto più avanzato dei calcoli che si possono eseguire con il pacchetto *calc*.

La definizione è la seguente:

```
\NewDocumentCommand\setfontsize{m O{1.2}}{%
    \fontsize{#1}{\fpeval{#2*#1}}\selectfont}
```

Come si vede dalla lista dei descrittori degli argomenti, il primo descrittore si riferisce ad un argomento obbligatorio (m: mandatory), mentre il secondo ad un argomento facoltativo (O: optional) con un valore di default pari a 1.2. La macro `\fpeval` serve per calcolare il valore dell'espressione contenuta nel suo argomento; nel nostro caso si tratta del prodotto (`*`) del ⟨*fattore*⟩ (il secondo argomento `#2`) con il ⟨*corpo*⟩ desiderato (il primo argomento `#1`).

[1.5]Il risultato che si ottiene è che questa frase è composta in corpo 9.5 mentre questa frase è composta in corpo 10.5. La composizione di questo capoverso è stata eseguita avendo specificato il comando `\setfontsize{10}[1.5]`, cioè usando lo stesso corpo usato lungo tutta la guida, ma con uno scartamento corrispondente al 50% in più del corpo, invece del solito 20%.

19.3 Ridefinizione di comandi già esistenti

Per ridefinire un comando già esistente, ammesso che il compositore sappia che cosa stia ridefinendo e ne capisca tutte le possibili conseguenze, si usa il comando `\renewcommand` con la sintassi seguente:

```
\renewcommand{⟨macro⟩}[⟨num-argomenti⟩][⟨default⟩]{⟨definizione⟩}
\renewcommand*{⟨macro⟩}[⟨num-argomenti⟩][⟨default⟩]{⟨definizione⟩}
```

I cui parametri hanno lo stesso significato che per `\newcommand` e al quale si può aggiungere l'asterisco come si può fare con `\newcommand`.

Il caso più frequente della ridefinizione di comandi esistenti riguarda il caso della modifica di comandi propri oppure di comandi definiti nei file di classe o di estensione.

In questi casi è *vietato* modificare i file originali, ma si procede come indicato nel paragrafo 6.7.

Va da sé che se si cerca di ridefinire un comando che non esiste, l'istruzione `\renewcommand` emette un perentorio messaggio d'errore e si rifiuta di procedere oltre.

19.3.1 Ridefinizione della parole fisse

Un uso abbastanza frequente consiste nel ridefinire le parole che compaiono durante l'esecuzione di comandi di sezionamento. Per esempio il comando `\listoffigures` esegue l'elenco delle figure come un indice a sé stante che viene introdotto da un titolo diverso a seconda della lingua usata. In italiano questo titolo è "Elenco delle figure"; questa stringa è contenuta nella definizione della macro `\listfigurename`. Per cambiare questo titolo, per esempio in "Lista delle figure", basta allora ridefinire la macro nel modo seguente:

```
\renewcommand*{\listfigurename}{Lista delle figure}
```

Naturalmente questo implica che il testo venga scritto in una sola lingua e che la ridefinizione sia eseguita dopo il comando di inizio del documento, quando la lingua di default è già stata specificata. Bisogna poi stare attenti a non cambiarla più; in particolare, se *babel* fosse chiamato con un elenco di lingue, perché il documento contiene citazioni in lingue diverse da quella di default, bisogna evitare accuratamente di usare il comando `\selectlanguage` per evitare di rieseguire la definizione dei comandi di default della lingua principale, il che renderebbe vana la ridefinizione appena illustrata. Se si deve scrivere davvero un documento in molte lingue con diversi elenchi delle figure scritti e intitolati in lingue diverse, allora *babel* consente di usare altri comandi più potenti, come `\addto`[5], che permettono di essere più efficaci nel gestire queste cose.

[5]Le parole contenute in apposite macro, come `\listfigurename`, che a loro volta sono contenute dentro una macro il cui nome è ottenuto dall'agglutinazione della parola `captions`

Questo avviso serve appunto a segnalare come anche una ridefinizione 'innocente' come quella proposta può causare risultati non previsti, anche se non sono dannosi per l'integrità di LaTeX. Il compositore che desidera usare `\renewcommand` deve sempre sapere veramente quello che sta facendo.

19.3.2 Ridefinizione del comando `\XeLaTeX`

Il comando `\XelaTeX` ha la prima "E" che appare capovolta o riflessa, a seconda di come la si guardi.

Usando i font OpenType non ci sono problemi ad usare questo tipo di "E", ma quei font possono essere usati solo con xelatex e lualatex.

Nello stesso tempo il nucleo di LaTeX non contiene una macro per scrivere XeLaTeX. Bisogna caricare il pacchetto *metalogo* che permette di usare una varietà di loghi, configurabili a seconda dei font usati.

Anche questo pacchetto deve distinguere come comporre il logo XeLaTeX a seconda che il programma di composizione sia pdflatex o uno degli altri due. Nel caso di pdflatex, che può usare solo font con 256 segni, l'apposito segno della "E" va in qualche modo simulato, perché nessun font Type 1 lo contiene. In quel pacchetto si usa il comando `\reflectbox` del pacchetto *graphicx* per scambiare la destra con la sinistra, vale a dire come riflettere la "E" in uno specchio verticale. Tutto bene finché non si usano caratteri inclinati, come il corsivo, perché la "E" inclinata e riflessa nello specchio appare inclinata nel verso sbagliato. Si confronti "E" e la sua immagine "Ǝ" se si usa un carattere diritto; con "*E*" la sua immagine è "*Ǝ*" se si usa un carattere inclinato.

La soluzione a questo problema consiste nel controllare se il carattere in uso è inclinato e nel ricorrere a una rotazione di 180° attorno al suo centro. In questo documento, composto con pdflatex il problema è stato risolto ridefinendo il comando `\XeLaTeX` nel modo seguente:

```
\renewcommand*\XeLaTeX{X\ifdim\fontdimen1\font=0pt\kern-0.15em\fi
  \lower.5ex\hbox{\rotatebox[origin=c]{180}{E}}}
```

In un secondo momento si è notato che il comando poteva presentare delle fragilità e si è preferito sostituire `\renewcommand` con `\DeclareRobustCommand` così da disporre di un comando sicuramente robusto; vedi il paragrafo 19.6. Nella macro il test `\ifdim` verifica quanto valga l'inclinazione del font corrente; per farlo bisogna ricorrere ai comandi di LaTeX descritti nella sua guida `fntguide.pdf` relativa ai font; essa è accessibile con il comando da terminale `texdoc fntguide`; la prima dimensione del font corrente indica l'inclinazione del font in punti tipografici riferiti ad un punto tipografico in orizzontale, e la si misura con quella

---

con il nome della lingua; per l'italiano, dunque, questa macro si chiama `\captionsitalian`. Il comando `\addto`, pertanto, riceve come primo argomento la macro `\captionsitalian` e come secondo argomento la ridefinizione di `\listfigurename`. Così facendo ogni volta che si specifica la lingua italiana, la macro che contiene il titolo dell'elenco delle figure viene ridefinito di nuovo, ma questa operazione ridondante è un male minore rispetto a quanto descritto nel testo se si eseguisse la ridefinizione senza aggiungerla alle altre macro per l'italiano.

espressione `\fontdimen1\font`. Se detta inclinazione è nulla bisogna arretrare il segno di una frazione di `em`, mentre se il font è obliquo detto arretramento non è necessaria; invece è comunque necessario abbassare la lettera ruotata, e lo di fa con il comando nativo di TEX `\lower` che abbassa la scatola indicata con un altro comando nativo `\hbox`. Procedere in questo modo serve per semplificare un poco il codice interno, ma il risultato è lo stesso di quello che si otterrebbe con il comando `\raisebox{-0.5ex}{...}`. Ora quando si usa il comando `\XeLaTeX` si ottiene XƎLATEX e in corsivo *XƎLATEX*; solo un occhio molto esercitato riconosce la piccola imperfezione dovuta al fatto che la "E" non è riflessa, ma è ruotata. D'altra parte con pdflatex non si può fare di meglio.

19.4 Ridefinizioni di comandi di sistema

La ridefinizione dei comandi di sistema è una operazione da evitare se non si sa esattamente quello che si sta facendo; si rischia di buttare all'aria tutto il funzionamento dei programmi di composizione. Si veda anche quanto si è scritto nel paragrafo 6.7.

19.4.1 I numeri romani maiuscoletti

Per eseguire la scrittura dei numeri romani minuscoli il sistema di default usa esternamente il comando `\roman`, usabile dal compositore, ma internamente il comando del compositore viene eseguito da `\@roman`; il comando comincia con il segno `@` che per il programma non è una lettera e quindi difficilmente il compositore può andare a 'disturbare' i comandi interni di sistema. Tuttavia il modo esiste, ed è descritto da Leslie Lamport in persona nel suo manuale. Bisogna usare il comando `\makeatletter`[6] prima di ridefinire un comando interno; questa istruzione cambia i codici interni che descrivono la funzione del segno @ in modo da poterlo usare come se fosse una lettera dell'alfabeto.

Questo risolve il primo problema. Ma per scrivere in maiuscoletto non basta specificare `\textsc` perché questa forma non è definita per tutte le famiglie e per tutte le serie. Bisogna accontentarsi di simulare la forma maiuscoletta con lettere maiuscole di corpo più piccolo.

La ridefinizione di `\@roman` diventa allora la seguente:

```
\makeatletter
\DeclareRobustCommand*{\simulatedSC}[1]{%
\hbox{$\relax$\fontsize{\sf@size}{\f@baselineskip}\selectfont#1}}%
\renewcommand*{\@roman}[1]{\simulatedSC{\@Roman#1}}%
```

Una spiegazione però è necessaria. I comandi `\sf@size` e `\f@baselineskip` sono i comandi interni che contengono rispettivamente il corpo degli indici primi e dello

[6] Se la nuova definizione viene inclusa nel proprio file di macro personali, avente estensione `.sty` che viene richiamato con il comando `\usepackage`, non è necessario servirsi del comando `\makeatletter`.

scartamento associati al font corrente; il primo dei due è definito solo dopo che sia stata scritta una formula, qui ridotta a nulla (`\relax`); essi sono documentati insieme ad altri comandi dello stesso genere nella già citata guida `fntguide` di cui si è già detto. Il comando `\fontsize` è quello di base per impostare un nuovo corpo e un nuovo scartamento; ma questi non diventano operativi se non dopo aver dato il comando esplicito `\selectfont`; infine il comando `\@Roman` è il comando interno per scrivere i numeri romani in lettere maiuscole. Ciò premesso la ridefinizione che stiamo cercando di fare richiede un comando `\simulatedSC` con il quale si ordina di scrivere con l'avanzamento di riga corrente, ma con il corpo degli indici primi; per limitare i suoi effetti e per essere sicuri che il numero romano, cadendo in fin di riga, non venga eventualmente diviso in sillabe, il tutto viene eseguito dentro una scatola orizzontale `\hbox` che successivamente viene usata come un unico oggetto, non come una stringa di lettere/cifre romane; usando questo comando la ridefinizione di `\@roman` viene eseguita in modo da prendere il numero romano scritto in lettere maiuscole e di stamparlo nel corpo degli indici primi. Qui si mostra come la pagina corrente possa essere scritta in numeri romani maiuscoli e in numeri romani maiuscoletti simulati in corpi diversi[7]:

Corpo 'normalsize': CDLXXII CDLXXII

Corpo 'Huge': CDLXXII CDLXXII

Ad esaminare bene il problema, tenendo conto che se si usa il pacchetto *hyperref*, che non è di bocca buona quando si definiscono o ridefiniscono comandi che lui tratta in modo particolare per creare i riferimenti ipertestuali, bisogna evitare di usare comandi personali nei punti in cui *hyperref* interviene, ci si accorge che la definizione suddetta può andare bene in contesti molto semplici, ma non in generale.

La soluzione generale non è quella di ridefinire il modo di comporre i numeri romani quando questi possono essere usati da *hyperref*, quindi tutte le volte che si maneggiano i numeri delle pagine. Bisogna inserire il comando per usare o per simulare il maiuscoletto solo nei punti in cui il numero viene effettivamente composto, quindi nelle testatine, nell'indice analitico, e nell'indice generale; inoltre il comando di simulazione del maiuscoletto potrebbe essere usato anche in altre circostanze, come, per esempio, nel comporre il nome del programma BibTeX, il cui logo prevede l'uso del maiuscoletto nelle seconde due lettere ib. Esistono infatti dei font che non solo non hanno il maiuscoletto nero, ma non hanno nemmeno il maiuscoletto *tout court*.

Allora per le testatine bisogna ridefinire i comandi per lo stile della pagina o delle pagine iniziali dove queste potrebbero essere numerate in numeri roma-

[7] Caricando la lingua greca, anche se non è la lingua principale, e il pacchetto *hyperref*, quanto spiegato non funziona, perché quel modulo per il greco e quel pacchetto modificano la definizione del comando `\@roman` in modo incompatibile.

ni minuscoli; per questo testo[8] è stato necessario ridefinire lo stile `headings`, chiamandolo `frontmatter` e lo stile `plain` chiamandolo `plainfrontmatter`. Semplicemente si sono copiate le definizioni di `plain` e di `headings` e si è aggiunto il comando `\textsc` per modificare la composizione del numero della pagina, indicato dal comando `\thepage`:

```
% Stile delle pagine iniziali con il numero di pagina in maiuscoletto
\def\ps@frontmatter{%
    \ps@headings
      \def\@evenhead{\textsc{\thepage}\hfil\slshape\leftmark}%
      \def\@oddhead{\slshape\rightmark\hfil\textsc{\thepage}}%
    }

\def\ps@plainfrontmatter{%
     \let\@mkboth\@gobbletwo
     \let\@oddhead\@empty
     \def\@oddfoot{\reset@font\hfil\textsc{\thepage}\hfil}%
     \let\@evenhead\@empty
     \let\@evenfoot\@oddfoot
    }
```

Per l'indice analitico non si è fatto nulla di particolare sul modo di comporlo, ma si sono modificati i comandi con i quali i pochissimi lemmi presenti nelle pagine iniziali sono stati indicizzati.

Per l'indice generale la cosa è stata più complessa: il comando che genera la riga per le sezioni a livello di paragrafo o di livello inferiore non presentano nessun problema perché sono scritti in un font di serie media. Il comando in questione è `\@dottedtocline`, che va modificato inserendo la dichiarazione per l'uso del maiuscoletto, come si vede nella terzultima riga del codice seguente:

```
\def\@dottedtocline#1#2#3#4#5{%
  \ifnum #1>\c@tocdepth \else
    \vskip \z@ \@plus.2\p@
    {\leftskip #2\relax \rightskip \@tocrmarg
     \parfillskip -\rightskip
     \parindent #2\relax\@afterindenttrue
     \interlinepenalty\@M
     \leavevmode
     \@tempdima #3\relax
     \advance\leftskip \@tempdima \null\nobreak\hskip -\leftskip
     {#4}\nobreak
     \leaders\hbox{$\m@th
        \mkern \@dotsep mu\hbox{.}\mkern \@dotsep
```

[8] In realtà tutte queste ridefinizioni erano state necessarie nelle prime edizioni di questo testo, quando nella *front matter* si usava la numerazione romana delle pagine. Successivamente, come è stato spiegato altrove, si è deciso di numerare tutte le pagine della guida con cifre arabe (ridefinendo i comandi `\frontmatter` e `\mainmatter`) e il problema è sparito del tutto.

```
        mu$}\hfill
    \nobreak
    \hb@xt@\@pnumwidth{\hfil\normalfont\scshape \normalcolor #5}%
    \par}%
\fi}
```

Questa operazione, tra l'altro semplicissima, potrebbe essere evitata usando il pacchetto *tocloft*, che serve per personalizzare il modo di comporre gli indici, purché i font da usare abbiano la serie e la forma desiderate.

Per la riga che serve per comporre nell'indice i riferimenti ai capitoli, che è composta con un font della serie nera, bisogna agire con un comando di simulazione; infatti il font Latin Modern usato in questo testo, non dispone del maiuscoletto nero. A nulla varrebbe ricorrere a pacchetti come *tocloft*, se poi il font da usare non contiene la serie o la forma giusta. Bisogna pertanto mettere le mani nei comandi interni. La stessa cosa che qui farò per comporre la voce relativa ai capitoli dovrebbe venire ripetuta per comporre le altre voci che appaiono in neretto, come per esempio le voci che riguardano le parti; siccome questo testo non è diviso in parti (è diviso in tomi, però), mi accontento di modificare solo le voci dei capitoli.

La riga che definisce il modo di comporre la voce dei capitoli si chiama `\l@chapter`; visto che il font usato in questo testo non contiene il maiuscoletto nero, bisogna usare la simulazione mediante il comando `\simulatedSC`, ma con apposite varianti rispetto a quanto esposto precedentemente:

```
\renewcommand*\l@chapter[2]{%
  \ifnum \c@tocdepth >\m@ne
    \addpenalty{-\@highpenalty}%
    \vskip 1.0em \@plus\p@
    \setlength\@tempdima{1.5em}%
    \begingroup
      \parindent \z@ \rightskip \@pnumwidth
      \parfillskip -\@pnumwidth
      \leavevmode \bfseries
      \advance\leftskip\@tempdima
      \hskip -\leftskip
      #1\nobreak\hfil \nobreak
      \hb@xt@\@pnumwidth{\hss\simulatedSC{#2}}\par
      \penalty\@highpenalty
    \endgroup
  \fi}
```

Ecco, ora il comando `\simulatedSC` deve svolgere il suo compito correttamente, e per questo deve agire solo sulle lettere minuscole, non sulle cifre o sulle lettere maiuscole; la definizione data precedentemente non fa queste distinzioni, quindi ci vogliono definizioni nuove. Innanzi tutto ridefiniamo il `\@roman` in modo che la ridefinizione venga usata solo nella parte principale e non nella parte iniziale del testo:

```
\renewcommand\mainmatter{%
  \cleardoublepage
  \@mainmattertrue
  \pagenumbering{arabic}%
  \def\@roman##1{\expandafter\simulatedSC
                 \expandafter{\romannumeral##1}}%
  }
```

e notiamo che invece di ricorrere a `\@Roman`, come fatto precedentemente, usiamo il comando primitivo `\romannumeral` il cui argomento, sia esso un numero o un contatore, viene espresso mediante la stringa di lettere minuscole che ne rappresentano la scrittura in numeri romani ma, attenzione, queste lettere non sono di categoria 11, come lo sono abitualmente le lettere lette dal programma di composizione dai file sorgente, ma sono di categoria 12 (segni analfabetici); quindi non ne possiamo controllare la categoria, visto che in questo modo lettere e cifre hanno la stessa categoria. Bisogna aggirare questo problema con mezzi che potrebbero ricorrere ai comandi di ε-TEX, oppure ai nuovi comandi avanzati di LaTeX 3, ma qui si mostra un modo per superare questo scoglio che ricorre a semplici applicazioni del linguaggio primitivo di TEX. Definiamo una nuova variabile booleana `lettere` per sapere se la stringa da convertire in maiuscoletto simulato sia formata da lettere, e impostiamo il valore di questa variabile mediante un comando che prende la stringa da convertire, ne controlla la prima (e forse unica) lettera, e se è una lettera minuscola imposta il valore 'vero' per la variabile booleana:

```
\newif\iflettere\letterefalse

\def\testlettere#1#2!{\count256=‘#1
    \ifnum\count256<‘a \letterefalse
    \else
        \ifnum\count256>‘z \letterefalse
        \else
          \letteretrue
        \fi
    \fi}
```

La definizione di `\testlettere` viene eseguita con un comando primitivo, perché usa argomenti delimitati; in questo caso la lista degli argomenti è delimitata a destra da un punto esclamativo; `\testlettere` riceve la stringa da convertire, ne estrae la prima lettera nell'argomento `#1`, e le altre (eventualmente nessuna) fino al delimitatore costituito dal punto esclamativo, le assegna all'argomento `#2`; si presuppone che la stringa da convertire sia formata solo da lettere; per fare le cose meglio bisognerebbe usare ricorsivamente questo comando su tutta la stringa e lo si potrebbe fare passandole di nuovo la stringa contenuta in `#2` ma, con questa riserva mentale, ci accontentiamo di supporre che la stringa da controllare sia formata solo da lettere e ne controlliamo solo la prima. Poi il valore

numerico dell'indirizzo interno della lettera contenuta in `#1` viene assegnata al contatore 256 (valore consentito da quando pdflatex contiene le estensioni di ε-TeX); questo si ottiene usando il 'backtick' (o virgoletta semplice aperta o accento grave, purtroppo assente dalla tastiera italiana, con la quale bisogna arrangiarsi immettendo il codice 96 con il tastierino numerico e il tasto Alt); il backtick è un comando primitivo di TeX che consente di estrarre il codice del carattere a cui esso è premesso; infatti poco dopo si usano `‘a` e `‘z` per avere i codici interni della lettera 'a' e della lettera 'z'. Ecco quindi che con un doppio test numerico si controlla se la lettera contenuta in `#1` è esterna all'intervallo delle lettere minuscole latine `a...z`, nel qual caso la variabile booleana `lettere` viene posta falsa, altrimenti è vera, cioè la lettera controllata è una lettera minuscola.

A questo punto possiamo definire `\simulatedSC` in modo che agisca solo sulle lettere minuscole:

```
\DeclareRobustCommand*{\simulatedSC}[1]{%
  \testlettere#1!\iflettere\textormath{%
  \hbox{$\relax$%
  \fontsize{\sf@size}{\f@baselineskip}\selectfont\uppercase{#1}}%
  }{%
  \scriptstyle\mathrm{#1}}%
  \else#1\fi}%
```

Il meccanismo è sempre lo stesso; si determina il corpo dei pedici di primo ordine in relazione al font in uso e con quello si compone in maiuscoletto simulato se si è in modo testo; si noterà però che a causa del comando `\testlettere` questa operazione è fatta solo se stiamo trattando una stringa di lettere minuscole (almeno se la prima lettera della stringa è una lettera minuscola) altrimenti la stringa viene lasciata inalterata.

Questa serie di definizioni, inserite nel file `MacroGuida.sty`, ci ha permesso di scrivere correttamente BibTeX, anche se la famiglia di font senza grazie non dispone del maiuscoletto; ci ha permesso (nelle edizioni precedenti di questa guida) di comporre correttamente l'indice generale, le testatine e i piedini della pagine iniziali di questo testo.

Tanto fumo per così poco? Ebbene sì, ma la cosa dipende principalmente dal fatto che i font Latin Modern sono privi di maiuscoletto nero. Se si fossero usati dei font non standard, per esempio attraverso il pacchetto *fourier*, contenenti il maiuscoletto nero, tutto ciò si sarebbe potuto evitare e si sarebbe potuto usare il pacchetto *tocloft* che con semplici comandi di impostazione avrebbe permesso di ottenere gli stessi risultati.

19.4.2 Comandi da eseguire solo per determinate lingue

Prima di continuare il commento su queste macro, vale la pena di sottolineare che raramente con il linguaggio primitivo, o con il mark-up di LaTeX, un problema di programmazione ha una sola soluzione; questi problemi hanno praticamente

sempre una moltitudine di soluzioni e tutto sta nel trovare quella più robusta, o quella più efficiente, o quella che, insomma, di volta in volta sembra la migliore sotto punti di vista diversi.

Componendo in inglese non si deve eseguire nessuna definizione strana della virgola; quindi se si vogliono inserire questi comandi in un file di macro personali, è meglio racchiuderli dentro gli argomenti del comando di `babel` `\iflanguage` che ha la seguente sintassi:

```
\iflanguage{⟨lingua⟩}{⟨vero⟩}{⟨falso⟩}
```

dove ⟨*lingua*⟩ è il nome dell'opzione passata a `babel` come lingua di default, mentre ⟨*vero*⟩ corrisponde alle azioni da eseguire se la lingua di default è quella specificata, altrimenti si eseguono le azioni indicate da ⟨*falso*⟩.

Tuttavia in realtà non c'è bisogno di fare niente, perché in inglese la virgola verrebbe usata come punteggiatura, quindi non sarebbe mai seguita da una cifra (se non nelle liste di valori numerici) e come separatore decimale verrebbe usato il punto il cui codice matematico non è stato cambiato; quindi scrivendo in inglese basterebbe avere le stesse attenzioni che si hanno in italiano, salvo usare il punto come separatore decimale.

19.4.3 Approfondimenti

Questi esempi ricorrono a comandi primitivi e ai codici di categoria matematici; non si tratta di concetti adatti ad una introduzione, ma non sono nemmeno concetti inaccessibili; il lettore che voglia approfondire deve esaminare e studiare il TEXbook, dove questi argomenti sono trattati in diversi capitoli; si veda qui anche il capitolo 21. Conviene anche documentarsi sul nucleo di LATEX, sebbene la documentazione sia piuttosto essenziale; la si legge dando da terminale il comando `texdoc source2e`.

19.4.4 La virgola intelligente

Un altro esempio usato in questo testo è costituito dalla ridefinizione della virgola. È noto che in tutte le lingue, tranne l'inglese, le norme ISO prescrivono come separatore decimale la virgola. In questo modo in tutte le altre lingue la virgola svolge due ruoli, ma solo in matematica. Quando si è in modo matematico il segno della virgola può essere dichiarato attivo e gli si può associare una definizione. Qui si è fatto esattamente così:[9]

[9]Attenzione con le virgolette `"`: se i comandi che seguono vengono emessi quando è già in vigore l'opzione *italian* di `babel`, l'interprete quando legge `"8000` segnala un errore; per convenienza e semplicità di scrittura sarebbe meglio sostituire in quella definizione la notazione esadecimale con quella ottale: `'100000`; più semplicemente ancora si può scrivere `\string"8000`. Si faccia anche attenzione al 'backtick' `‘`, che non è un accento ma il carattere ASCII 96; la scrittura di questo carattere può essere difficile con la tastiera italiana e con sistema operativo Windows; in un laptop che non abbia il tastierino numerico separato (quasi tutti) bisogna affidarsi alle prestazioni dello *shell editor*; in un PC con la tastiera munita di tastierino

19.4.4.1 La virgola intelligente che riconosce lo spazio

Definiamo una semplice soluzione al problema della virgola che capisca da sola se in matematica deve comportarsi da segno di punteggiatura o da separatore decimale. Basta definire il glifo della virgola come attivo solo in matematica e dargli una definizione per riconoscere se nel testo sorgente è o non è seguito da uno spazio.

```
\DeclareMathSymbol{\virgola}{\mathpunct}{letters}{"3B}
\DeclareMathSymbol{\virgoladecimale}{\mathord}{letters}{"3B}
\AtBeginDocument{\mathcode`\,="8000}
{\catcode `,=\active \gdef,{\futurelet\let@token\m@thcomma}}

\def\m@thcomma{%
   \ifx\let@token\@sptoken \virgola
   \else \virgoladecimale\fi
}
```

Precisamente si sono definite due entità matematiche: `\virgola` che rappresenta un segno matematico di punteggiatura e `\virgoladecimale` che rappresenta la virgola decimale; la differenza risiede nel codice interno che il carattere `"3B` (la virgola) riceve attraverso i comandi `\DeclareMathSymbol`, visto che nel primo caso è specificato appartenente alla categoria `\mathpunct` (punteggiatura matematica), mentre nel secondo caso esso è definito come `\mathord`, cioè come un normale simbolo o una variabile matematica. La specifica `letters` in entrambi i casi dice che il segno deve essere tratto dalla stessa polizza di caratteri dai quali si traggono i simboli letterali (il font corsivo matematico).

Le definizioni del codice attivo e della sua definizione diventano attive solo all'inizio del documento mediante `\AtBeginDocument`; vi si dice che il codice matematico deve essere posto a `"8000`, lo speciale codice che definisce attivo un carattere in modo matematico. Gli si assegna poi attraverso il comando primitivo `\gdef` una definizione globale nella quale esso deve assegnare a `\let@token` non il significato del token successivo, che sarebbe `\m@thcomma`, ma quello del token che incontrerà ancora dopo; allora e solo allora potrà eseguire `\m@thcomma`. Questo a sua volta è definito in modo da esaminare il tipo di token il cui significato è stato attribuito a `\let@token`; se questo è uno spazio (`\@sptoken`) usa la virgola di punteggiatura, altrimenti usa la virgola come simbolo. I comandi `\ifx`, `\else` e `\fi` servono appunto per eseguire questi comandi condizionali. Sta ora al compositore lasciare uno spazio dopo la virgola (in matematica) solo quando questa rappresenta un segno di punteggiatura, mentre non deve lasciare nessuno spazio se deve comportarsi da separatore decimale.

Il risultato di questo comando è visibile in ogni numero decimale fratto scritto in questo testo[10], ma qui è opportuno confrontare bene che cosa si ottiene se in

numerico separato dal resto, il backtick può venire introdotto premendo sul tastierino il numero 96 mentre si tiene premuto il tasto Alt. Purtroppo la tastiera italiana non è di grande aiuto; per fortuna non capita spesso di dover usare il backtick.

[10] Sebbene in effetti in questo testo si sia usata una delle definizioni descritte qui di seguito.

matematica si scrive `f(x,y)` oppure `f(x,␣y)`: $f(x,y)$ nel primo caso, ma $f(x, y)$ nel secondo. La differenza è piccola, ma si vede benissimo.

19.4.4.2 La virgola intelligente che riconosce se è seguita da una cifra

Tuttavia ci si può domandare se per caso non sia possibile fare a meno dell'inserimento di uno spazio dopo la virgola, quando questa ha il significato di segno di interpunzione; sì, è possibile, ma bisogna definire la virgola attiva in modo un poco più complesso:

```
\def\m@thcomma{\let\@tempB\virgola
\@tfor\@tempA:=0123456789\do{%
\expandafter\ifx\@tempA\let@token\let\@tempB\virgoladecimale
\@break@tfor\fi}\@tempB}
```

che si basa su una macro interna del nucleo di LaTeX, `\@tfor`, la quale ripete un certo insieme di comandi tante volte quanti sono i token che seguono l'indicazione `:=`. Nel nostro caso questi token sono costituiti dalle dieci cifre decimali; l'argomento del comando `\do`[11] viene ripetuto assegnando successivamente uno dei token della lista alla macro `\@tempA`, quindi viene ripetuto al massimo dieci volte. Se il token seguente la virgola nel testo sorgente, memorizzato nella sequenza `\let@token`, è uguale al token via via memorizzato nella macro `\@tempA`, cioè è un cifra decimale, allora il valore di default di `\@tempB`, inizializzato con la `\virgola` interpuntiva, viene cambiato nel valore attributo alla virgola decimale, `\virgoladecimale`, e al termine della macro ripetitiva viene eseguito il comando `\@tempB`. Gli altri comandi presenti nel corpo della macro ripetitiva, come `\expandafter` o `\@break@tfor`, servono per scopi di programmazione particolari: `\expandafter` è strumentale per eseguire il confronto fra il contenuto di una macro e un token implicito e, per poterlo fare, bisogna prima sviluppare la macro; `\@break@tfor` serve per interrompere le iterazioni del comando ripetitivo non appena si è scoperto che il token implicito è una cifra.

Usando questa definizione per `\m@thcomma`, invece di quella precedentemente indicata, il "tastierista" non deve ricordarsi di inserire spazi dopo la virgola, tranne in un unico caso, quando dovesse scrivere una lista di numeri, come in:

```
...␣per␣ogni␣$i=1,␣2,␣3,\dots,n$
```

che produce "... per ogni $i = 1, 2, 3, \ldots, n$". Si vede chiaramente che prima di n c'è lo stesso spazio che precede le cifre, anche se non c'è lo spazio nel codice sorgente. Se si fosse omesso lo spazio fra la virgola e la cifra successiva si sarebbe ottenuto invece: "... per ogni $i = 1,2,3, \ldots, n$", che è palesemente errato.

[11] La sequenza di controllo `\do` sembra un comando perché è seguita da una coppia di graffe che contengono ciò che si deve eseguire; in realtà `\do` è solo un segno per delimitare a destra il primo argomento di `\@tfor`, cioè le dieci cifre, delimitate a sinistra da `:=`.

19.4.4.3 La virgola intelligente che riconosce il codice di categoria del carattere successivo

Ci sarebbe qualche altro modo per evitare di dover ricorrere a comandi interni del nucleo di LaTeX o a magie del linguaggio primitivo di TeX? Sì, si potrebbe fare, anche se è necessario comunque ricorrere ai comandi primitivi `\ifx`[12], `\else` e `\fi`, abbastanza chiari nel loro semplice inglese anche a chi non conosce la programmazione condizionale. Bisogna definire un comando ausiliario:

```
\def\vic@code{%
\ifx\let@token0\number1\else
\ifx\let@token1\number1\else
\ifx\let@token2\number1\else
\ifx\let@token3\number1\else
\ifx\let@token4\number1\else
\ifx\let@token5\number1\else
\ifx\let@token6\number1\else
\ifx\let@token7\number1\else
\ifx\let@token8\number1\else
\ifx\let@token9\number1\else
\number0\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi}
```

e poi definire:

```
\def\m@thcomma{\ifnum\vic@code>0\relax
  \virgoladecimale\else\virgola\fi}
```

Da un punto di vista pratico il risultato è lo stesso di quello ottenuto con la seconda definizione; ora però il codice è più facilmente comprensibile ed è anche più efficiente. I dieci confronti fra `\let@token`, (che, ricordiamo, è il carattere implicito equivalente al segno che segue immediatamente la virgola nel codice sorgente) con le dieci cifre decimali sembra più lungo di quello eseguito con il ciclo `\@tfor`, ma è molto efficiente. Il motivo è estremamente tecnico e lo si tralascia; tuttavia non è difficile capire che per gestire il ciclo `\@tfor` il motore di composizione deve sviluppare ripetutamente diverse macro che non appaiono nella versione scritta dall'utente, ma che sono nascoste nel nucleo di LaTeX. Questa terza soluzione, invece, ricorre solo a comandi primitivi e non deve sviluppare niente; inoltre uno dei pregi dei comandi condizionali è che sono costruiti in modo tale da eliminare i rami "falsi" dalla lista di token da scandire in fase di input, per cui alla fine il motore di composizione si ritrova solamente a inserire nella lista dei token che `\ifnum` deve esaminare, solo uno 0 o un 1, avendo eliminato tutti i rami risultati "falsi" di quella lunga serie di `\ifx`.

[12] Il significato di `\ifx` è diverso da quello di `\if`; entrambi esistono, ma il secondo sviluppa gli oggetti da confrontare prima di confrontarli, mentre il primo confronta i loro significati senza svilupparli.

19.4.4.4 La virgola intelligente che riconosce se il carattere successivo ricade nella lista dei caratteri ASCII delle cifre

Sembra strano, ma non c'è nessun comando primitivo né una macro che permetta di stabilire direttamente se un token sia una cifra o un altro segno. Tuttavia si potrebbe ricorrere a questa quarta soluzione che ricorre sempre a comandi primitivi e a *sporchi trucchi* del linguaggio primitivo TEX; richiede, però, conoscenze più approfondite:

```
\def\m@thcomma#1{%
\unless\ifcat\noexpand\let@token*
  \virgola
  \expandafter\expandafter\expandafter
      \ifcsname\expandafter\@gobble\string#1\endcsname
    \virgola
  \else
    \ifnum‘#1<‘0\relax
      \virgola
    \else
      \ifnum‘#1>‘9\relax
        \virgola
      \else
        \virgoladecimale
      \fi
    \fi
  \fi
\fi#1}
```

Questa soluzione richiede solo quattro comandi condizionali; il primo `\ifcat` serve per verificare il codice di categoria di due token[13]; perciò `\ifcat` controlla se il token memorizzato in `\let@token` abbia lo stesso codice di categoria dell'asterisco; questo è un segno 'qualunque', dell'*alfabeto* ASCII, che ha lo stesso codice di categoria delle cifre; `\let@token` invece potrebbe contenere una cifra (codice di categoria 12), uno spazio (codice di categoria 10) un carattere attivo o un comando (codice di categoria 13), una lettera (codice di categoria 11) un segno analfabetico come l'asterisco, le parentesi, i segni 'più' o 'meno', eccetera (codice di categoria 12 come le cifre), un altro carattere implicito[14].

Allora se il codice di categoria *non* è 12 (quello dell'asterisco) la virgola non è certo seguita da una cifra e deve essere resa con la virgola di punteggiatura `\virgola`; altrimenti bisogna riconoscere se si tratti di un altro carattere implicito e lo si fa con quella lunga serie di comandi intercalati con sequenze di `\expandafter` che servono per 'invertire' l'ordine di esecuzione dei vari comandi:

[13] Siccome non sappiamo se `\let@token` contenga un carattere normale o un comando sviluppabile, lo facciamo precedere da `\noexpand` così da evitarne lo sviluppo.

[14] Cioè potrebbe essere un'altra sequenza di controllo a cui è stato assegnato il significato di un carattere mediante `\let`.

prima `\string`, poi `\@gobble` poi `\ifcsname`; in questo modo si prende l'argomento `#1`, lo si converte in una stringa di caratteri, gli si toglie il primo carattere, e si verifica se quello che resta può essere il nome di una sequenza di controllo; le cifre non sono assegnate con caratteri impliciti, ma potrebbe essere un carattere (come, per esempio, `\infty`) che potrebbe essere stato reso equivalente mediante `\let` o un altro comando primitivo legato a un carattere matematico; in questo caso non si può procedere al riconoscimento di una cifra con la "magia" del segno ‘ che si può applicare ai caratteri espliciti; bisogna poi riconoscere se si tratti di una cifra. Si noti che il primo token diverso da uno spazio che segue la virgola è stato assorbito dal comando `\m@thcomma` come suo argomento; il primo test controlla se il codice ASCII di questo token (`‘#1`) precede il codice ASCII dello zero (`‘0`): se il test è vero non si tratta di una cifra e ci vuole la virgola di punteggiatura. Bisogna però fare un altro test dello stesso genere per verificare se il codice ASCII del token segue il codice ASCII della cifra nove (`‘9`), nel qual caso ci vuole la virgola interpuntiva, altrimenti siamo in presenza di una cifra e si usa la `\virgoladecimale`. Tenuto conto che i comandi condizionali primitivi eliminano dal flusso di ingresso i rami falsi, questo algoritmo è quello che consente la massima efficienza fra i quattro esposti.

19.4.5 Lo spazio per il numero nelle liste delle tabelle e delle figure

Un altro esempio, tratto dalle piccole modifiche eseguite per la composizione di questo testo, è un comando per dare un formato alle righe degli elenchi delle figure e delle tabelle. Il comando di formato di queste righe è specificato nel file della classe, nel nostro caso di `book.cls`; vi si legge

```
\newcommand*\l@figure{\@dottedtocline{1}{1.5em}{2.3em}}
...
\let\l@table\l@figure
```

Si osserva innanzi tutto che il comando per la formattazione delle righe della lista delle tabelle, `\l@table` viene reso equivalente alla definizione appena data al comando per la formattazione delle figure, `\l@figure`. Se quindi cambiamo il secondo, dobbiamo ricordarci di eseguire nuovamente l'equivalenza dei comandi.

Per quel che riguarda la formattazione di queste righe vediamo che il comando `\l@figure` si affida all'esecuzione del comando `\@dottedtocline`, il cui nome ricorda il fatto che queste righe degli indici sono di solito fornite della fila di puntini che guidano l'occhio dal titolo al numero della pagina. I parametri che questo comando riceve sono in realtà cinque, ma gli altri gli vengono passati dagli altri comandi che servono per recuperare e stampare le informazioni da inserire nell'indice, cioè il titolo (nel nostro caso la didascalia (breve) della figura o della tabella) e il numero della pagina. Ma nel listato di sopra i tre parametri hanno i significati seguenti: (*a*) il primo dice che la riga deve essere composta come quella dei paragrafi (livello 1); (*b*) il secondo rappresenta il rientro della riga; si

nota infatti che la riga dei paragrafi nell'indice di questo testo è rientrata, quella dei sotto paragrafi è rientrata di più, eccetera; (*c*) il terzo parametro specifica lo spazio da destinare al numero specifico della figura o della tabella oltre allo spazio che divide questo numero dall'inizio del titolo riportato nell'indice. Ecco, questo terzo valore va benissimo se il numero del capitolo non è superiore a 9 o se il numero della figura o della tabella non è superiore anch'esso a 9, cioè se almeno uno dei due sia di una sola cifra. In questo testo alcuni dei capitoli con un numero di due cifre contengono più di nove figure, e non resta nessuno spazio fra il numero e il titolo; il che sta evidentemente male. Lo spazio di `2.3em` è insufficiente, quindi bisogna aumentarlo di quanto basta per metterci un'altra cifra; `1em` è largo praticamente come una lettera 'M', e un cifra è larga circa la metà (la cifra 1 graficamente sembra più stretta, ma in realtà insieme agli spazi che la contornano occupa lo stesso spazio delle altre; è voluto, perché così le tabelle di numeri risultano incolonnate anche se contengono delle cifre 1). Basta allora ridefinire il comando di sistema così:

```
\renewcommand*\l@figure{\@dottedtocline{1}{1.5em}{2.8em}}
\let\l@table\l@figure
```

ricordando di inserire queste ridefinizioni in un file di macro personali (a causa del segno `@`) e di non dimenticare l'equivalenza fra `\l@table` e `\l@figure`.

19.5 Esiste già o non esiste ancora il comando?

Talvolta si vuole essere sicuri che un comando sia disponibile ma non si sa se esso sia già stato definito in qualche pacchetto di estensione. Allora si può usare il comando `\providecommand` già descritto all'inizio di questo capitolo.

Si ricorre spesso a questo tipo di definizione mediante `\providecommand` nel proprio file di macro personali, cosicché il comando definito attraverso questa istruzione viene sempre definito se il comando non è già stato definito da altri, altrimenti questa definizione viene tranquillamente ignorata e viene usata la definizione già esistente. Logicamente perché questo funzioni come desiderato, è necessario che il proprio file di macro personali sia sempre invocato per ultimo.

Nel preambolo di questo testo, per esempio, si è provveduto il comando `\ohm` nel modo seguente

```
\providecommand*{\ohm}{\textormath{\textohm}{\mathrm{\Omega}}}
```

dove `\textormath` è un comando fornito dal pacchetto *babel* per specificare due azioni diverse da eseguire in modo testo o in modo matematico; in modo matematico si è specificato di usare il font tondo diritto, per ovviare al caso in cui si sia deciso di usare le lettere greche maiuscole inclinate nelle normali espressioni matematiche.

19.6 Definizione di comandi robusti

Un comando si dice robusto se esegue quel che deve eseguire in qualunque circostanza; ciò non è sempre vero, specialmente se esso contiene nella sua definizione dei comandi condizionali e se costituisce o può costituire l'argomento di un altro comando.

È prudente, allora, definire comandi robusti confezionati con cautele particolari, i cui dettagli sono troppo tecnici da spiegare qui. Il tutto viene eseguito mediante il comando `\DeclareRobustCommand` che può venire usato solo nel preambolo o, bene inteso, anche nei file di classe o di estensione, che comunque vengono letti solo nel preambolo. La sua sintassi è identica a quella di `\newcommand`; bisogna stare attenti che esso non verifica se il comando da definire esista già o sia davvero un comando nuovo, quindi questa 'dichiarazione' deve venire usata con molta cautela da programmatori esperti; tuttavia se si fosse definito un comando con `\newcommand` e questo avesse passato i controlli di definibilità, ma nell'uso risultasse fragile, allora non sarebbe un problema cambiare il comando `\newcommand` con `\DeclareRobustCommand` e tutto procederebbe come desiderato con la certezza di non avere pasticciato con comandi già esistenti. In queste circostanze, se si è caricato nel preambolo o nei file di estensione, prima della definizione della macro eseguita con `\newcommand` e la si vuole trasformare in modo che sia robusta, diventa comodo premettere al comando l'operatore `\robustify` che rende robusto il comando ricorrendo ai comandi estesi presenti nel motore di composizione in uso già da almeno una decina d'anni. Nell'esempio del paragrafo 19.3.2 invece di sostituire `\newcommand` con `\DeclareRobustCommand`, si sarebbe potuto premettere al `\newcommand` il comando`\robustify` (sempre che nel preambolo o nel file di macro si fosse caricato il pacchetto *etoolbox*.

Nel capitolo 29 viene descritto il comando `\protect`; esso serve per rendere robusti i comandi fragili; il comando `\DeclareRobustCommand` agisce attraverso `\protect` ma passa anche per l'intermediario di un nuovo comando, apparentemente con lo stesso nome di quello che costituisce il suo argomento, ma prolungato con uno spazio che fa parte integrante del suo nome: se si vuole rendere robusto il comando `\pippo`, `\DeclareRobustCommand` protegge mediante `\protect` il comando `\pippo␣` e a questo assegna la definizione specificata. Questo aiuta a rendere robusti i comandi quando essi vengono scritti nei file ausiliari; quando questi file vengono riletti in una esecuzione successiva, tutto quello che viene letto ripassa attraverso il processo di 'tokenizzazione', e lo spazio letto dal file ridiventa un normale spazio privo di significato quando segue il nome di un comando. Ingegnoso, ma talvolta i neofiti non riescono a rendersi conto di questo passaggio. A questo proposito merita leggere nel capitolo 30 il comando `\ShowCommand` che è in grado di mostrare nella console e nel file `.log` il significato anche di macro protette.

19.7 Le definizioni di comandi nuovi con LaTeX 3

Continuiamo con la virgola intelligente, ma usando il linguaggio LaTeX 3.

19.7.1 La virgola intelligente ottenuta con LaTeX3

Come ho già detto, i comandi di LaTeX 3 (che si chiamano *funzioni*) permettono di fare cose che con LaTeX 2_ε, sono complicate. Vediamo come si possono modificare le definizioni viste nei paragrafi precedenti usando queste funzioni.

La virgola intelligente che riconosce dal suo codice di categoria se il carattere successiv

Nel paragrafo 19.4.4.3 si è visto come analizzando il codice di categoria i test da eseguire diventano più complicati da scrivere, ma più semplici da eseguire. Qui vediamo come si può risolvere questo caso senza eseguire esplicitamente l'analisi della categoria. Precisamente il comando da ridefinire sarebbe lo stesso della voce che segue.

La virgola intelligente che riconosce direttamente se il carattere successivo è una cifra

Esiste una funzione di LaTeX 3 che permette di eseguire dei test sul codice di carattere esaminandone il token, ma trascurando il codice di categoria; basta usare la macro `\IfDigit` definita con il nuovo linguaggio per decidere se usare la virgola interpuntiva o la virgola matematica. Il codice da usare (da inserire fra i codici `\ExplSyntaxOn` e `\ExplSyntaxOff` presumibilmente nel preambolo del documento, o in un file di macro personali insieme ad altre macro definite in modo simile) è il seguente:

```
\ProvideExpandableDocumentCommand\IfDigit{m m m}%
{\bool_if:nTF{\str_compare_p:eNe{#1} < {0} ||
              \str_compare_p:eNe{#1} > {9}}{#3}{#2}
}
```

la cui sintassi è la seguente:

`\IfDigit{`⟨*test*⟩`}{`⟨*codice vero*⟩`}{`⟨*codice falso*⟩`}`

Il test è eseguito controllando lo stato booleano di una espressione logica formata da due confronti "letterali" collegati dal connettore logico `+||+` che è l'operatore OR: il primo confronto "letterale" controlla se lo sviluppo del primo argomento è lessicograficamente precedente alla cifra 0, e il secondo, sempre lessicograficamente, controlla se viene dopo la cifra 9; se l'uno o l'altro confronto è vero, il carattere ottenuto dallo sviluppo di `\let@token` ha un codice di carattere diverso da quelli delle cifre che cadono nell'intervallo (0–9), quindi non è una cifra; allora viene eseguito il ⟨*codice falso*⟩, invece se il test è positivo viene eseguito il *codice vero*.
Perciò la definizione di `\m@thcomme` seguente fa tutto quello che serve:

```
\def\m@thcomma#1{%
\unless\ifcat\noexpand\let@token*%
```

```
  \virgola
\else
  \IfDigit{#1}{\virgoladecimale}{\virgola}
\fi#1}
```

Qualche commento è utile, visto che questa definizione ha una prima parte piuttosto complessa, come nel paragrafo 19.4.4.4. Abbiamo già commentato le prime sette righe, quelle che precedono il comando `\IfDigit`; sembrerebbe che il linguaggio LaTeX 3 contenga delle funzioni per controllare il codice di categoria e il codice di carattere, ma finora le prove che si sono fatte non ci hanno permesso di ridurre la complessità di quelle sette righe; qui ripetiamo che il primo test permette di evitare tutto il resto se il carattere che segue la virgola *non* è un carattere di categoria matematica 'other' come lo è l'asterisco; il secondo test verifica se quanto segue è una control sequence o una macro generica, e se lo fosse inserirebbe una virgola interpuntiva saltando di controllare tutto il resto. Finalmente, essendo rimasta la possibilità che il carattere successivo possa essere una cifra, controlla se tale carattere sia effettivamente una cifra araba compresa nell'intervallo (0–9) e se lo è, inserisce una virgola decimale. Guardando la definizione di `\IfDigit` si vede proprio che la virgola interpuntiva viene usata quando il test è falso, cioè quando il segno che segue la virgola *non* è una cifra.
Il codice seguente:

```
$f(x,y)\quad f(x, y)\quad 123,456\quad 123, 456\quad 23,?456$
```

si traduce infatti in: $f(x,y)$ $f(x, y)$ $123{,}456$ $123, 456$ $23, ?456$ Va da sé che gli utenti devono stare molto attenti quando scrivono matematica, perché si tratta di una 'lingua' priva di ridondanza ed è difficilissimo scoprire dal contesto gli errori di scrittura o il significato di quanto si sarebbe voluto esprimere.

19.7.2 Cicli ripetitivi

Spesso si devono ripetere dei cicli di operazioni un numero di volte fisso o che dipende dal contesto; Fra i comandi nativi di LaTeX 2_ε c'e il comando `\@tfor...\do{...}` che, come abbiamo visto può servire per fare ciclicamente una certa operazione analizzando una lista di token. Il linguaggio LaTeX 3 consente di fare cicli di operazioni con test diversi che consistono in espressioni booleane complesse.

Le macro che in questa guida sono state usate per aggiungere in coda un certo numero di pagine bianche per rendere il numero totale di pagine un multiplo intero del numero di facciate di una segnatura, sono costruite usando l'uno o l'altro comando definito come segue.

Nel file `MacroGuida.sty`, che contiene le macro locali per la composizione di questa guida, fra i delimitatori `\ExplSyntaxOn` e `\ExplSyntaxOff` (necessari per usare il linguaggio LaTeX 3) ci sono un certo numero di definizioni fra le quali `\fpdowhile`, `\fpwhiledo` e `\Mod`:

```
\ProvideExpandableDocumentCommand\fpdowhile{m m}{%
   \fp_do_while:nn{#1}{#2}}
\ProvideExpandableDocumentCommand\fpwhiledo{m m}{%
   \fp_while_do:nn{#1}{#2}}
\ProvideExpandableDocumentCommand\Mod{m m}%
 {\inteval{\int_mod:nn{#1}{#2}}}
```

le cui sintassi sono le seguenti:

```
\fpdowhile{⟨espressione logica⟩}{⟨ciclo⟩}
\fowhiledo{⟨espressione logica⟩}{⟨ciclo⟩}
\Mod{⟨numero⟩}{⟨modulo⟩}
```

I primi due comandi ripetono i comandi contenuti nell'argomento ⟨*ciclo*⟩ finché l'⟨*espressione logica*⟩ è vera. Sebbene sembrino fare la stessa cosa la loro differenza è essenziale: il primo verifica lo stato 'vero' o 'falso' dell'espressione logica dopo aver usato i comandi del ciclo una prima volta; il secondo invece comincia ad eseguire i comandi del ciclo dopo aver controllato che l'espressione logica sia vera. È evidente che prima di eseguire `\fpdowhile` e `\fpwhiledo` l'espressione logica deve essere impostata al valore 'vero' e durante l'esecuzione dei comandi del ciclo deve essere modificato qualcosa da cui dipende l'espressione logica, in modo che questa assuma il valore 'falso', altrimenti si instaura un ciclo infinito che diventa difficilissimo arrestare, a meno che non si fermi da solo per esaurimento della memoria di lavoro del programma di composizione; i risultati sarebbero disastrosi talvolta anche per la stessa installazione del sistema TeX.

Il comando `\Mod`, invece, calcola il risultato dell'espressione di aritmetica modulare $m \bmod n$ oppure $m \pmod n$, cioè il resto della divisione intera fra i numeri interi m ed n. Questo tipo di operazione è quella che eseguiamo normalmente per separare un numero di ore in giorni e ore del giorno: 60 ore corrispondono a 2 giorni e 12 ore; il numero 12 è il risultato di `\Mod{60}{24}`.

Con queste macro disponibili, le pagine da aggiungere in coda al documento, aumentate del numero di pagine iniziali della *front matter*, nel caso che siano numerate con una numerazione diversa da quella usata nella parti centrale e finale (*main matter* e *back matter*) del documento, dopo aver assegnato alla macro`\Segnatura` il numero di facciate di ciascuna segnatura si possono ottenere con i comandi

```
\def\Segnatura{⟨numero facciate⟩}
\clearpage
\pagestyle{empty}
\addtocounter{page}{⟨facciate iniziali⟩}
\fpwhiledo{\Mod\value{page}}{\Segnatura!=1}{\null\newpage}
```

inseriti subito prima di `\end{document}`. Si noti il costrutto `!=` che significa 'non uguale', visto che l'operatore logico `!` è l'operatore NOT.

Sembra strano che il ciclo si fermi quando il test corrisponde ad numero di pagine 'modulo `\Segnatura`' pari a 1. Ma a pensarci bene dopo ogni esecuzione dei comandi del ciclo, il numero di pagina punta a quella nuova, che non è da emettere perché la pagina nuova sarebbe successiva a quella che completava una segnatura. Ovviamente il ⟨*numero facciate*⟩ di ogni segnatura dipende dalla scelta dell'utente con la passione per la legatoria, che si rilega il documento da solo[15], oppure di quell'utente che fornisce alla tipografia le segnature già impostate, dopo aver richiesto quante facciate devono essere contenute in ciascuna segnatura. Il numero di ⟨*facciate iniziali*⟩ potrebbe essere determinato automaticamente se l'utente si ridefinisse i comandi `\frontmatter` oppure `\mainmatter`, ma fa sicuramente prima a leggerne direttamente il numero nella schermata di anteprima del visualizzatore associato al suo editor.

I questa guida, in cui non si usano i numeri romani per la numerazione delle pagine iniziali, il loro numero è semplicemente nullo.

19.8 Definizione di un nuovo ambiente

Per definire un nuovo ambiente si ricorre al comando `\newenvironment` che, prima di agire, verifica che l'ambiente da definire non sia già definito; se lo fosse emetterebbe un messaggio d'errore e la compilazione verrebbe interrotta. Il comando ha la seguente sintassi:

```
\newenvironment{⟨ambiente⟩}[⟨num_arg⟩][⟨default⟩]%
                {⟨comandi di apertura⟩}{⟨comandi di chiusura⟩}
```

dove ⟨*ambiente*⟩ è il nome dell'ambiente che si vuole definire.

L'ambiente che viene definito può ricevere *solo in apertura* un certo numero ⟨*num_arg*⟩ di argomenti il primo dei quali è facoltativo quando viene specificato anche⟨*default*⟩; esso va trattato come il primo argomento facoltativo definito con `\newcommand`. Con la stessa sintassi esiste il comando `\renewenvironment` che serve per ridefinire un ambiente già definito.

[15] I millanta moduli che fanno parte del sistema TeX non sono capaci di tenere conto dell'*abbondanza*; questa parola indica di quanto deve essere spostata verso l'esterno la griglia di stampa nei fogli delle segnature, affinché le griglie di stampa in ogni pagina sembrino impostate tutte correttamente, nonostante la curvatura del fogli nella sua piega centrale; a seconda dello spessore della carta questa abbondanza deve aumentare da zero per il foglio centrale ad un massimo per il foglio più esterno. Benché i fogli abbiano uno spessore che si aggira attorno al decimo di millimetro, una segnatura di 8 facciate richiede due fogli, quindi l'abbondanza del foglio esterno ammonta a circa 2 decimi di millimetro ed è del tutto trascurabile. Ma per una segnatura di 64 facciate, occorrono 16 fogli, e quello esterno compie una semicirconferenza di diametro corrispondente ai fogli piegati al suo interno, quindi di diametro pari a circa 3 mm; l'abbondanza del foglio esterno dovrebbe tenere conto anche della lunghezza della semicirconferenza, quindi dovrebbe essere di circa 2,5 mm per ogni griglia di stampa delle sue quattro facciate. Perciò al legatore in propio si consiglia di mantenere le segnature formate da 2 o al massimo tre fogli, corrispondenti a 8 o 12 facciate, per le quali la necessaria abbondanza sarebbe trascurabile.

Seguono poi i ⟨*comandi di apertura*⟩ da eseguire all'inizio dell'ambiente, il quali fanno uso esplicito degli argomenti eventualmente presenti, obbligatori o facoltativi; infine vengono definiti i ⟨*comandi di chiusura*⟩ da eseguire appena prima di chiudere l'ambiente.

In realtà definire un ⟨*ambiente*⟩ equivale a definire due comandi uno per l'apertura e l'altro per la chiusura, come se si fossero definiti i due comandi seguenti:

```
\newcommand{\⟨ambiente⟩}[⟨num_arg⟩][⟨default⟩]{⟨comandi di apertura⟩}
\newcommand{\end⟨ambiente⟩}{⟨comandi di chiusura⟩}
```

I comandi `\begin` e `\end` con i quali si aprono e si chiudono gli ambienti eseguono delle funzioni importanti fra le quali la costituzione di un gruppo, cosicché tutto quello che si fa dentro l'ambiente rimane locale e circoscritto solo e soltanto a quell'ambiente; il comando `\end`, oltre a chiudere il gruppo, provvede a verificare che si stia chiudendo l'ultimo ambiente che si era aperto; in caso contrario viene emesso un messaggio d'errore.

19.8.1 Gli ambienti *medaglione* e *sintassi*

Il fatto dei due comandi distinti per l'apertura e la chiusura di un ambiente può essere sfruttato nella definizione di nuovi ambienti; per esempio, in questo documento il file `MacroGuida.sty`, che contiene le definizioni utili alla composizione di questo testo, si definisce l'ambiente *sintassi* che provvede a stampare la sintassi dei vari comandi racchiudendo il suo contenuto in un rettangolo che fa da cornice. In questo modo si è pensato di definire l'ambiente *medaglione*[16] che incornicia il suo contenuto e l'ambiente *sintassi* che espone il medaglione incorniciato ma separato dal testo che lo precede e che lo segue mediante uno spazio verticale uguale a quello che si ha nei testi in display; in particolare si richiede che il testo sia giustificato solo a sinistra; le due definizioni sono le seguenti:

```
\newenvironment{medaglione}[1][\linewidth]{\setbox0\vbox\bgroup
       \hsize=\dimexpr#1-2\fboxsep-2\fboxrule\relax
\noindent}{\par\egroup\setbox0\vbox{\unvbox0}%
       \framebox{\box0}}
%
\newenvironment{sintassi}{\flushleft\medaglione}
                    {\endmedaglione\endflushleft}
```

L'interpretazione delle definizioni per l'ambiente *sintassi* è abbastanza semplice: in apertura esegue le istruzioni per comporre un testo in display giustificato solo a sinistra e poi esegue le operazioni per l'apertura di un medaglione; in chiusura

[16] Nella pagina 780 c'è un altro esempio simile, l'ambiente *riquadro*; merita osservare le differenze. Qui si descrive una definizione che era stata preparata nelle prime versioni di questo testo e che ricorre a comandi primitivi del linguaggio TEX; è una occasione per mostrare l'uso di tali comandi.

esegue le operazioni per terminare il medaglione e quelle per terminare il testo in display giustificato a sinistra.

> I comandi per la definizione del *medaglione* sono più complicati e richiedono la conoscenza approfondita dei comandi primitivi. Si osservi innanzi tutto che l'ambiente accetta un argomento facoltativo il cui valore in questo esempio vale `0.8\linewidth`. Questo può essere utile per incorniciare e mettere in evidenza anche testi che non abbiano riferimento con la sintassi dei comandi, come si è fatto con questo capoverso.

A parte l'argomento facoltativo, in sostanza i comandi di apertura dicono che con `\setbox` si deve riempire un registro di tipo `box`, quello numerato 'zero' con una scatola verticale `\vbox` la cui apertura è indicata con il comando `\bgroup`; in questa scatola verticale si compone del testo usando una giustezza `\hsize` impostato al valore corrente del primo argomento `#1` esplicitamente specificato o il suo valore di default; ma si diminuisce questo valore di quanto basta per tenere conto dello spazio fra la cornice e il suo contenuto, valore che è conservato dentro il registro di tipo 'lunghezza' chiamato `\fboxsep`, e dello spessore del filetto conservato dentro il registro `\fboxrule`; inoltre si ordina di comporre senza il rientro dei capoversi mediante `\noindent`.

Nei comandi di chiusura bisogna accertarsi che il capoverso costituito dal contenuto del medaglione sia effettivamente terminato ordinandone la terminazione con il comando `\par` e si termina il gruppo che chiude la scatola verticale con `\egroup`; si riassegna poi il contenuto della scatola 'zero' ancora alla scatola 'zero' estraendo le singole righe, ovverosia disfacendo la scatola 'zero' con il comando `\unvbox`; questo serve per essere sicuri che le righe siano convenientemente spaziate; spesso questa operazione è inutile, ma non guasta; infine si usa questa scatola con `\box` come argomento di `\framebox` al fine di incorniciarla.

19.8.2 Definizione di un ambiente per due figure

Talvolta si devono esporre figure di dimensioni relative piccoline, meno larghe di mezza griglia di stampa, o, almeno, la cui somma finale sia inferiore alla larghezza della griglia. Sembrerebbe utile metterle fianco a fianco, ciascuna con la sua didascalia.

Non si possono usare le funzionalità del pacchetto *subfigure*, perché questo serve per inserire diverse piccole figure *correlate* dentro un unico ambiente flottante, ciascuna dotata di una sua "sotto didascalia" numerata con lettere, e subordinata alla numerazione della didascalia principale. Utilissimo pacchetto, ma serve per una funzionalità diversa.

Si vorrebbe realizzare un ambiente che, appunto, metta due immagini le le cui larghezze non devono superare la larghezza della gabbia, ciascuna con la sua didascalia composta con la stessa giustezza dell'immagine ridimensionata in modo che abbiano la stessa altezza.

Il codice che segue funziona abbastanza bene:

```
1 \newenvironment{FIGURA}[2]{%                                  apertura
2 % #1 := file della figura di sinistra
3 % #2 := file della figura di destra
4 \def\strip{\csuse{strip@pt}}
5 \dimen8=1pt
6 \dimen6=0.4755\textwidth
7 \setbox0=\hbox{\includegraphics[width=\dimen6]{#1}}
8 \setbox2=\hbox{\includegraphics[width=\dimen6]{#2}}
9 \dimen4=\dimexpr(\ht0+\ht2)/2\relax
10 \edef\factuno{\csuse{strip@pt}\dimexpr\dimen4*\dimen8/\ht0}%
11 \edef\factdue{\csuse{strip@pt}\dimexpr\dimen4*\dimen8/\ht2}%
12 \dimen0=\factuno\dimen6
13 \dimen2=\factdue\dimen6
14   \def\primafigura{\begin{minipage}[t]{\dimen0}%
15   \resizebox{\hsize}{!}{\box0}}%
16   \def\secondafigura{\end{minipage}\hfill
17   \begin{minipage}[t]{\dimen2}%
18   \resizebox{\hsize}{!}{\box2}}%
19   \begin{figure}[!htb]
20 }{%                                                            chiusura
21   \end{minipage}%
22 \end{figure}
23 }
```

Esso definisce l'ambiente *FIGURA* il cui comando di apertura richiede due argomenti, come indicato nelle righe di commento: il primo corrisponde al nome del file che contiene la figura da inserire nella parte di sinistra, e il secondo quello della figura di destra. Nella riga 4 i due nomi vengono salvati in due macro temporanee, valide solo dentro l'ambiente stesso. Nelle righe da 5 a 8 si usano quattro registri di dimensione, numerati $0, 2, 4, 6$ e 8; secondo una convenzione che risale a Knuth stesso, i primi dieci registri possono essere usati come registri temporanei se hanno numeri pari; non è necessario usare solo i pari, perché tanto all'uscita dell'ambiente essi vengono ristabiliti con lo stesso valore che avevano all'apertura dell'ambiente.

Nei registri scatola 0 e 2 si inseriscono con `\includegraphics` le due figure, scalate in modo che abbiano la stesa larghezza pari al valore conservato nel registro 6, in modo da poterne rilevare le altezze al fine di determinare i fattori di scala necessari per portarle alla stessa altezza. Queste altezze si estraggono dalle due scatole mediante il comando nativo di TEX `\ht` (*height*, altezza); nel registro dimensionale 6 si era messo un valore un poco più piccolo della giustezza della gabbia di stampa; invece nel registro dimensionale 4 si mette la media delle due altezze misurate.

Siccome il sistema TEX usa i valori fratti solo come moltiplicatori delle unità di misura, va bene che si lavori con le grandezze dimensionali, ma dobbiamo

ricordare che i valori numerici che si possono estrarre da quei registri sono in punti tipografici, non in millimetri o altre unità. Ecco allora che i due fattori di scala `\factuno` e `\factdue` (numeri fratti adimensionati) si determinano ciascuno prendendo l'altezza media dal registro 4 dividendola per l'altezza naturale, in modo che quando si scalano le due figure ciascuna per il suo fattore di scala esse risultano della stessa altezza, pari alla media delle altezze iniziali.

Ma questi fattori di scala servono per scalare le due figure sia in altezza sia in larghezza; ecco allora che possiamo riutilizzare i registri dimensionali 0 e 2 per conservare la larghezza scalata con cui dimensionare le due figure da affiancare; ciò viene fatto nelle righe 15 e 18.

Ma per rendere più agevole il lavoro dell'utente, è meglio definire i due comandi `\primafigura` e `\secondafigura` usando i quali tutte quelle operazioni vengano svolte in automatico dietro le quinte cosicché l'utente può usare la seguente semplice sintassi:

```
\begin{FIGURA}{⟨Foto1⟩}{⟨Foto2⟩}
\primafigura
    \caption{⟨didascalia1⟩}\label\label{⟨label1⟩}
\secondafigura
    \caption{⟨didascalia2⟩}\label\label{⟨label2⟩}
\end{FIGURA}
```

Il lettore attento scopre da solo, prima ancora di cominciare ad usare questo ambiente e questa sintassi, che il calcolo dei fattori di scala produce effetti accettabili se le figure inizialmente non hanno coefficienti di forma *molto* diversi: se una è più larga che alta e l'altra è più alta che larga, i fattori di scala possono portare a larghezze tali che una occupa quasi tutta la larghezza disponibile e l'altra risulti tanto striminzita che lo spazio sottostante non possa contenere la didascalia. Tuttavia, in un caso normale, si vedano le figure 19.1 e 19.2 per apprezzare il risultato.

19.8.3 Commenti sugli ambienti

La persona che si avvicina per la prima volta al contenuto di questo capitolo può restare sconcertata anche da questo linguaggio strano, costituito dai comandi primitivi di TeX che, tra l'altro, non sono mai citati o descritti in questo testo. È vero, ma questo testo è introduttivo, non esaustivo; il lettore si accontenti di questo primo assaggio e poi si documenti convenientemente nel TeXbook se vuole andare oltre questi primi passi.

Nello stesso tempo si capisce bene a che cosa possano servire le macro; per eseguire nell'ordine giusto tutta una serie di operazioni; se si dovessero specificare ogni volta che si deve fare una certa operazione costringerebbero a scrivere e a riscrivere sempre le stesse cose con il pericolo di aggiungere errori ogni volta che si ripetono queste scritture.

Figura 19.1: Una piccola cucina

Figura 19.2: Mansarda con un letto a due piazze

19.9 La ridefinizione di ambienti esistenti

La ridefinizione di ambienti già esistenti si esegue con `\renewenvironment` che usa la stessa sintassi di `\newenvironment`. Viene scambiato solo il tipo di test: se l'ambiente da ridefinire non esiste, viene emesso un messaggio di errore e la compilazione viene interrotta.

Per gli ambienti non esiste un comando simile a `\providecommand` ma, in base a ciò che è stato detto a proposito dei nuovi ambienti, non dovrebbe essere difficile usare due volte il comando `\providecommand` per definire l'apertura e la chiusura di un ambiente; purché tutto ciò sia fatto nel proprio file di macro personali. Se si usa il pacchetto *xparse*, invece, sono disponibili anche i comandi per definire, ridefinire e provvedere gli ambienti che occorrono; vale la pena di notare che gli ambienti definiti mediante i comandi di *xparse* accettano argomenti obbligatori e facoltativi che possono venire usati anche nei comandi di chiusura; questa è un caratteristica importante, perché con i comandi normali di LaTeX si può usare un solo argomento facoltativo e tutti gli argomenti possono essere usati solo nei comandi di apertura.

Non è nemmeno difficile copiare dai file di classe o di estensione le definizioni di ambienti già esistenti travasando le definizioni nel proprio file di macro personali, per poi separare le due definizioni di apertura e di chiusura in due distinti comandi `\renewcommand` dentro alle cui definizioni apportare tutte le modifiche che si credono opportune.

19.9.1 Ridefinizione dell'ambiente *theindex*

Qui si porterà un esempio abbastanza articolato relativo alla modifica di un ambiente di sistema, la modifica dell'ambiente *theindex* per la composizione dell'indice analitico.

Si vorrebbe far sì che l'indice sia sempre composto a due colonne, ma facendolo precedere da una spiegazione composta a piena pagina. Vorremmo anche che l'indice analitico figurasse anche nell'indice generale. La prima cosa da fare è quella di esplorare l'archivio CTAN per vedere se il problema è già stato risolto. Qui supporremo di non aver trovato nulla che vada bene per il nostro caso[17] e quindi, rimboccandoci le maniche, ma esercitando la nostra creatività, ci accingiamo a modificare la definizione esistente.

Bisogna quindi copiare la definizione dell'ambiente *theindex* dal file di classe che stiamo usando per comporre il nostro documento nel nostro file di macro personali `mymacros.sty`. Per la classe *book* copieremo pertanto il codice seguente:

```
\newenvironment{theindex}
   {\if@twocolumn
      \@restonecolfalse
    \else
      \@restonecoltrue
    \fi
    \twocolumn[\@makeschapterhead{\indexname}]%
    \@mkboth{\MakeUppercase\indexname}%
            {\MakeUppercase\indexname}%
    \thispagestyle{plain}\parindent\z@
    \parskip\z@ \@plus .3\p@\relax
    \columnseprule \z@
    \columnsep 35\p@
    \let\item\@idxitem}
   {\if@restonecol\onecolumn\else\clearpage\fi}
%
\newcommand\@idxitem{\par\hangindent 40\p@}
\newcommand\subitem{\@idxitem \hspace*{20\p@}}
\newcommand\subsubitem{\@idxitem \hspace*{30\p@}}
\newcommand\indexspace{\par
                \vskip 10\p@ \@plus5\p@ \@minus3\p@\relax}
```

Le ultime definizioni servono per comporre le singole voci, in particolare per inserire i debiti rientri al fine di mettere in evidenza i tre livelli di voci disponibili. L'ultima definizione, quella di `\indexspace` serve per inserire uno spazio verticale nell'elenco composto ogni volta che si inizia una serie di voci con una nuova lettera dell'alfabeto.

[17] In realtà esiste il pacchetto *imakeidx* che fa molto di più di quello che si descrive qui a titolo di esempio.

Per poter scrivere su due colonne dopo aver cominciato a scrivere a piena pagina è conveniente aver caricato il pacchetto *multicol*; potremmo anche inserire un test per verificare se il pacchetto è stato davvero già caricato, ma per semplicità ometteremo questo controllo.

Allora modifichiamo la definizione dell'ambiente, in particolare la definizione dei comandi di apertura: (*a*) in modo da inserire l'ordine di scrivere nell'indice generale il nome dell'indice analitico e in modo da *non* usare il comando `\twocolumn` ma l'ambiente *multicols* fornito dal pacchetto *multicol*. Lasciamo stare gli ultimi quattro comandi come sono, per cui qui non li ripetiamo. Ovviamente cambiamo `\newenvironment` in `\renewenvironment` visto che stiamo ridefinendo un ambiente esistente. Inoltre il nuovo ambiente accetta un argomento facoltativo come primo ed unico argomento (quindi da racchiudere fra parentesi quadre) che contiene il testo esplicativo per l'uso dell'indice analitico.

```
\renewenvironment{theindex}[1][]
   {\if@twocolumn
       \@restonecolfalse
     \else
        \@restonecoltrue
      \fi
      \@makeschapterhead{\indexname}%
      \@mkboth{\MakeUppercase\indexname}%
              {\MakeUppercase\indexname}%
      \addcontentsline{toc}{chapter}{\indexname}%
      \thispagestyle{plain}%
      \parindent\z@
      \parskip\z@ \@plus .3\p@\relax
      \columnseprule \z@
      \columnsep 35\p@
      \let\item\@idxitem
      %               Testo di spiegazione
      #1
      \par
      \vspacs{3\baselineskip}%
      \multicols{2}}%              Fine comandi di apertura
     {%                            Inizio comandi di chiusura
      \endmulticols
      \if@restonecol\onecolumn\else\clearpage\fi
   }
```

Qualche parola di spiegazione non guasta. Il test iniziale eseguito mediante il comando `\if@twocolumn` appare all'inizio di ogni definizione di comandi o di ambienti che comincino un nuovo capitolo numerato o non numerato. Servono per stabilire se dopo la chiusura dell'ambiente sia o non sia necessario ripristinare la composizione a una colonna; l'ultimo test, nei comandi di chiusura eseguito

con il comando `\if@restonecol`, serve appunto per ripristinare la composizione ad una colonna nel caso che sia necessario.

Il comando `\@makeschaperhead` è il comando con il quale viene composto il titolo di un capitolo non numerato; in questo caso il titolo del capitolo è contenuto nella variabile `\indexname` che *babel* carica con il nome giusto a seconda della lingua in uso: 'Indice analitico', 'Index', ...

Il comando `\mkboth`, seguito dai suoi due argomenti serve per impostare il contenuto delle due testatine, prima viene indicato il contenuto della testatina di sinistra, poi di quella di destra. Il contenuto è sempre il nome in lingua corrente dell'indice analitico, ma reso in lettere maiuscole attraverso il comando `\MakeUppercase`.

Il comando `\addcontentsline`, come dice il nome, serve per aggiungere una riga alla 'table of contents', cioè all'indice generale. I suoi tre argomenti indicano rispettivamente la sigla della 'table of contents', `toc`, il tipo di voce da aggiungere a questa lista; in questo caso si tratta di una voce corrispondente al titolo di un capitolo; segue poi il titolo del capitolo che in questo caso continua ad essere il nome dell'indice analitico nella lingua corrente.

Il comando `\thispagestyle` specifica lo stile compositivo di questa prima pagina di questo nuovo capitolo; in questo caso si tratta dello stile `plain` che non contiene la testatina, ma contiene il numero della pagina nel piedino. Gli altri stili definiti di default sono `empty`, senza testatina né piedino, `headings` con piedino vuoto e testatina il cui contenuto viene specificato automaticamente dai comandi `\chapter` oppure `\section`; in particolare nella testatina di sinistra compare il titolo del capitolo e in quella di destra il titolo del paragrafo corrente all'inizio della pagina. Infine `myheadings` è simile a `headings` solo che il contenuto delle due testatine deve essere fissato manualmente dal compositore specificando di volta in volta entrambe le testatine oppure solo quella di destra attraverso i comandi `\markboth` o `\markright`.

La scrittura `\parindent \z@` serve per specificare che il rientro del capoverso deve essere di zero punti; questo valore è memorizzato nella variabile di sistema `\z@` che equivale alla scrittura 0 pt.

La successiva espressione `\parskip\z@ \@plus .3\p@\relax` serve per specificare un pochino di gomma elastica da inserire fra un capoverso e l'altro; in pratica fra una voce e l'altra; questo minimo di allungamento consente di giustificare verticalmente le colonne della composizione a due colonne.

La specificazione successiva `\columnseprule \z@` specifica che lo spessore del filetto verticale che separa le colonne è di zero punti (quindi il filetto non c'è perché non lo si vede). Invece `\columnsep 35\p@` dice che la separazione fra le due colonne vale 35 pt, quindi una decina di millimetri.

Al comando `\item`, che abitualmente viene usato nelle liste, qui viene assegnata un'altra definizione, quella che è stata attribuita al comando `\@idxitem` e che non è stata modificata nell'eseguire questa nuova definizione dell'ambiente *theindex*.

Segue poi il testo di spiegazione, che qui è stato indicato col numero dell'unico argomento facoltativo passato al comando di apertura dell'ambiente; volendo

si potrebbe fare un test per verificare se l'argomento `#1` (che di default non contiene nulla) contenga un po' di testo; se si carica il pacchetto *etoolbox* questo test si potrebbe fare semplicemente sostituendo le righe

```
#1
\par
```

con la riga:

```
\ifcsvoid{#1}{}{#1\par}
```

Finita la spiegazione il comando `\par` assicura che il capoverso contenente le spiegazioni sia veramente terminato; il successivo comando di spaziatura verticale dato con `\vspace` inserisce una spaziatura di tre righe di testo; infatti `\baselineskip` indica appunto lo scartamento delle righe del testo corrente.

I comandi di apertura finiscono con l'apertura dell'ambiente *multicols* al quale viene specificato di comporre su due colonne; non è stata usata la sintassi normale per aprire l'ambiente, perché siamo sicuri di chiuderlo nei comandi di chiusura, come in effetti facciamo eseguendo il comando `\endmulticols`.

Questo esempio è istruttivo sotto molti aspetti; dà una buona idea di ciò che viene eseguito dietro le quinte quando si dà un comando qualunque; dice quanto sia comodo che le macro possano fare tutto quello che fanno, così che con un semplice comando di mark-up si possano sistematicamente e uniformemente eseguire le stesse operazioni compositive; quanto lavoro si risparmi nell'usare macro ogni volta che sia possibile, e quindi quanto sia opportuno che, in base al testo che si sta componendo, il file `mymacros.sty` contenga proprio le definizioni delle macro personali che servono per comporre le strutture lessicali che più frequentemente compaiono nel testo.

19.9.2 Ridefinizione dell'ambiente *thebibliography*

L'ambiente *thebibliography* serve per comporre una bibliografia a mano, come descritto nel capitolo 11. Questo ambiente non inserisce il "capitolo" della bibliografia nell'indice generale. Usando i pacchetti che estendono la gestione della bibliografia all'uso dei database bibliografici, viene generato un comando del tipo `\printbibliography`, che accetta alcune opzioni fra le quali si trova anche l'opzione *intoc* (o simile, a seconda del pacchetto usato) che permette di aggiungere all'indice anche la voce per la bibliografia.

Ma qui ci si riferisce all'ambiente di default della bibliografia, in particolare quello della classe *book*. Come si è fatto per l'indice analitico, si copia dal file *book.cls* la definizione dell'ambiente *thebibliography* e la si incolla nel file di macro personali. Il codice originale è il seguente:

```
\newenvironment{thebibliography}[1]
     {\chapter*{\bibname}%
      \@mkboth{\MakeUppercase\bibname}{\MakeUppercase\bibname}%
      \list{\@biblabel{\@arabic\c@enumiv}}%
```

```
          {\settowidth\labelwidth{\@biblabel{#1}}%
           \leftmargin\labelwidth
           \advance\leftmargin\labelsep
           \@openbib@code
           \usecounter{enumiv}%
           \let\p@enumiv\@empty
           \renewcommand\theenumiv{\@arabic\c@enumiv}}%
     \sloppy
     \clubpenalty4000
     \@clubpenalty \clubpenalty
     \widowpenalty4000%
     \sfcode`\.\@m}
    {\def\@noitemerr
      {\@latex@warning{Empty `thebibliography' environment}}%
     \endlist}
\newcommand\newblock{\hskip .11em\@plus.33em\@minus.07em}
\let\@openbib@code\@empty
```

Il lettore ha qui un ottimo esempio per studiare i comandi interni dell'ambiente. Noti che le ultime due righe (che non verranno modificate) sono comandi utili per dare attuazione all'opzione *openbib* della classe *book*.

Per ottenere l'inserimento della voce in lingua contenuta nella macro `\bibname` impostata da *babel*, si deve ridefinire l'ambiente aggiungendo `\addcontentsline`; lavorando, quindi, sul codice copiato nel file delle macro personali, si deve cambiare `\newenvironment` con `\renewenvironment` e si deve aggiungere la nuova istruzione completa dei suoi tre argomenti obbligatori. Si ottiene, perciò, il codice seguente:

```
\renewenvironment{thebibliography}[1]
     {\chapter*{\bibname}%
      \@mkboth{\MakeUppercase\bibname}{\MakeUppercase\bibname}%
      \addcontentsline{toc}{chapter}{\bibname}% <---   aggiunta
      \list{\@biblabel{\@arabic\c@enumiv}}%
           {\settowidth\labelwidth{\@biblabel{#1}}%
            \leftmargin\labelwidth
            \advance\leftmargin\labelsep
            \@openbib@code
            \usecounter{enumiv}%
            \let\p@enumiv\@empty
            \renewcommand\theenumiv{\@arabic\c@enumiv}}%
      \sloppy
      \clubpenalty4000
      \@clubpenalty \clubpenalty
      \widowpenalty4000%
      \sfcode`\.\@m}%                              Fine apertura
     {%                                           Inizio chiusura
```

```
\def\@noitemerr
 {\@latex@warning{Empty ‘thebibliography’ environment}}%
\endlist}
```

Vale la pena osservare la presenza alla fine dei comandi di apertura dell'istruzione `\sfcode‘\.\@m`; il comando `\sfcode` serve per impostare il valore dello *space factor* del punto fermo indicato con la forma che si riferisce al suo codice interno: "`‘\.`"; gli viene assegnato il valore 1000, contenuto nella sequenza `\@m`. Il valore 1000 esprime in millesimi il valore del fattore che si usa per allargare lo spazio interparola dopo il carattere indicato dal primo argomento di `\sfcode`. È uno dei valori impostati dal comando `\frenchspacing`, di cui si parla nel paragrafo 29.2.1, nel quale si è osservato che nella bibliografia, dove le abbreviazioni sono numerose lo spazio dopo i segni di interpunzione di fine periodo sono resi uguali. Precisamente l'ambiente per la bibliografia cambia *solo* lo spazio dopo il punto fermo, ma non lo cambia dopo i punti esclamativo e interrogativo, che verosimilmente non compaiono mai nelle bibliografie.

19.10 Situazioni particolari

19.10.1 Le linee guida

Le *linee guida* sono quelle linee continue oppure linee di puntini o di altri simboli che guidano l'occhio da una parte all'altra della pagina come in questa riga:

Inizio ..... Fine

In inglese le linee guida si chiamano *leaders*.

LaTeX contiene già le definizioni di alcune linee guida: `\hrulefill` e `\dotfill`; il primo comando produce una linea continua appoggiata alla base della riga di stampa_____come in questa riga formata da una linea spessa 0,4 pt; agendo a livello di comandi primitivi è possibile ridefinire il comando o definirne un altro con una riga più spessa e/o non adagiata sulla linea di base; l'altro comando produce una fila di puntini ..... come in questa riga.

È possibile definire o ridefinire i comandi che generano le linee guida, ma ovviamente è sconsigliabile ridefinire i comandi del nucleo di LaTeX indicati sopra, perché LaTeX li usa dentro altri comandi e la loro modifica ne potrebbe alterare in modo irreparabile il loro funzionamento; perciò qui faremmo esempi che usano `\newcommand` per essere sicuri di non ridefinire nulla di preesistente.

Ogni linea guida è creata con una serie di comandi secondo la sintassi seguente:

```
\leaders⟨scatola⟩\hskip⟨gomma⟩
\cleaders⟨scatola⟩\hskip⟨gomma⟩
\xleaders⟨scatola⟩\hskip⟨gomma⟩
```

La comprensione dei termini usati in queste definizioni è facilitata se si rivedono questi concetti nel capitolo 29.

Brevemente, in questo contesto per ⟨*scatola*⟩ si intende sia una scatola vera e propria da riempire con quello che si vuole, per esempio un rettangolo nero (una scatola riempita di nero); il comando `\hskip` serve per produrre uno spostamento orizzontale; perché la linea guida sia arbitrariamente estensibile lo spostamento deve essere *elastico*, ed è per questo che viene chiamato ⟨*gomma*⟩, *skip* in inglese; questa ⟨*gomma*⟩ è una speciale lunghezza che è formata da tre termini:

1. la lunghezza naturale;
2. l'allungamento il cui ammontare è preceduto dalla parola chiave `plus`, sostituita nel nucleo di LaTeX, dalla macro `\@plus`;
3. l'accorciamento il cui ammontare è preceduto dalla parola chiave `minus`, sostituita nel nucleo di LaTeX, dalla macro `\@minus`.

Tutte le lunghezze definite con i comandi di LaTeX sono in effetti delle *gomme*. Per esempio la lunghezza definita come segue:

```
\newlength{\lamialunghezza}
\setlength{\lamialunghezza}{10pt plus 5pt minus 3pt}
```

definisce una lunghezza elastica (gomma) con il nome `\lamialunghezza`, la cui lunghezza naturale è di 10 pt, il cui allungamento vale 5 pt e il cui accorciamento vale 3 pt: insomma, una lunghezza che si può estendere da 7 pt a 15 pt a seconda delle necessità.

Tanto per focalizzare meglio la natura della gomma, lo spazio interparola è una lunghezza elastica che ha una sua larghezza naturale pari a quella della lettera 'e' nel font corrente e un allungamento e un accorciamento entrambi più piccoli della larghezza naturale; TeX tiene conto di quanto lo spazio interparola presente fra le parole di una riga si debba allargare o restringere per la sua giustificazione al fine di determinare la "bellezza" della riga, o meglio la sua "bruttezza"; in base ad elaborati calcoli eseguiti sui valori di allungamento e accorciamento effettivamente usati, TeX determina un livello di bruttezza da confrontare con il valore della `\tolerance` o della `\pretolerance` al fine di accettare o non accettare quella particolare riga, oppure per decidere se eseguire la cesura in fin di riga, oppure se emettere un messaggio che avvisa l'operatore che in quel capoverso c'è una riga troppo vuota (underfull hbox) o sporgente fuori dal margine (overfull hbox). Si tratta di concetti molto tecnici, ma è meglio averne una conoscenza almeno superficiale.

Tornando alle nostre linee guida, la ⟨*gomma*⟩ da usare deve avere una componente di allungamento infinita se vogliamo che i nostri leader si possano allungare a piacere. Infatti le definizioni delle due linee guida predefinite sono:

```
\def\hrulefill{\leavevmode\leaders\hrule\hfill\kern\z@}
```

```
\def\dotfill{\leavevmode
             \cleaders \hb@xt@ 0.44em{\hss.\hss}\hfill\kern\z@}
```

Il comando `\leavevmode` assicura che il comando venga usato in modo orizzontale. Il comando `\kern\z@` assicura che dopo la linea guida si prosegua senza ulteriori spazi e senza andare a capo; il comando `\hfill` indica di inserire uno spazio dato da una gomma con una lunghezza naturale nulla e un allungamento infinito.

Nella definizione di `\hrulefill` compare la scatola nera, il rettangolo nero, `\hrule`, con una larghezza naturale che non influisce sulla lunghezza della linea guida, con altezza pari a 0,4 pt e profondità nulla; il suo effetto è quindi di produrre una linea guida lunga quanto occorre, ma spessa solo 0,4 pt appoggiata alla linea di base.

Nella definizione di `\dotfill` compare il codice `\hb@xt@ 0.44em` il cui argomento vale `{\hss.\hss}`. Il primo comando è scritto in modo sintetico ed equivale (quasi) al comando LaTeX: `\makebox[0.44em]`. L'argomento di questa scatola è un punto fermo e da due gomme `\hss` di spazio orizzontale di lunghezza naturale nulla ma di allungamento e accorciamento infiniti. Il tutto è (quasi) equivalente al comando LaTeX `\makebox[0.44em]{.}` e il punto è centrato dentro la scatola larga 0,44 pt.

Tuttavia sopra si sono indicati tre tipi di linee guida, uno introdotto da `\leaders`, il secondo da `\cleaders`, il terzo da `\xleaders`; che differenza c'è fra questi tre tipi di linee guida?

1. `\leaders` introduce i suoi puntini definiti mediante una scatola vera, non con un rettangolo nero, come se fossero su una griglia fissa, per cui da una riga all'altra, come succede per esempio in un indice generale, i puntini sono sempre incolonnati, anche se questo comporta un eventuale spazio bianco all'inizio e alla fine, ma che l'incolonnamento dei puntini impedisce di riempire.

2. `\cleaders` introduce i suoi puntini centrando la linea di puntini senza preoccuparsi che siano incolonnati; va bene in una linea guida isolata di testo ma non in un indice generale dove le linee guida, se ci sono, è meglio che abbiano i puntini incolonnati.

3. analogamente `\xleaders` introduce i suoi puntini allargando un poco lo spazio fra di loro in modo da estendere la linea guida così che non solo il primo e l'ultimo spazio siano uguali, ma anche lo spazio restante sia distribuito fra i puntini presenti.

Si considerino le seguenti tre definizioni di tre diverse linee guida:

```
\newcommand*\puntini{\leaders\hbox to 10pt{\hss.\hss}\hfill\kern0pt}
\newcommand*\cpuntini{\cleaders\hbox to 10pt{\hss.\hss}\hfill\kern0pt}
\newcommand*\xpuntini{\xleaders\hbox to 10pt{\hss.\hss}\hfill\kern0pt}
```

e si inseriscano le linee guida così definite in una scatola larga 99 pt inserita fra due parole. Siccome ogni scatola-puntino delle linee guida è larga 10 pt, nello spazio di 99 pt ce ne stanno 9; i restanti 9 pt vengono distribuiti diversamente fra l'inizio e la fine a seconda del tipo di linea guida usata:

```
\leaders:  Prima . . . . . Dopo
\cleaders:  Prima . . . . . Dopo
\xleaders:  Prima . . . . . Dopo
```

Si noti anche che nelle definizioni del nucleo di LaTeX la grandezza della scatola-puntino è definita in termini di 'em', cioè di una unità di misura che dipende dal font usato; quindi anche se `\leaders` dovrebbe essere in grado di assicurare l'incolonnamento dei puntini, questo non può succedere se i capitoli sono composti in neretto e i paragrafi in serie media. Questo è il motivo per il quale nell'indice generale di questo testo (e nelle classi standard) le righe che riguardano i capitoli sono senza linee guida. Se le avessero, succederebbe quello che appare nel seguente esempio:

1 Primo capitolo 1
1.1 Primo paragrafo 1
1.2 Secondo paragrafo 2

Si vede chiaramente la mancanza dell'incolonnamento fra i puntini della riga del capitolo e quella dei paragrafi; si nota anche la presenza di spazi diversi fra le linee guida e il contenuto che loro "uniscono" visivamente. Queste disuniformità, comunque, diminuiscono via via che le scatole-puntino vengono accorciate e i puntini effettivamente inseriti nelle linee guida sono più densi.

Va anche da sé che al posto dei puntini si potrebbero indicare degli altri simboli testuali, matematici o decorativi.

19.10.2 Controllo della posizione di grandi oggetti

In questo testo lo schema del layout delle pagine è mostrato sia a partire dal verso dell'occhiello, sia nelle pagine 170 e seguente, sia nelle pagine 536 e seguente.

Nello stesso modo si potrebbe spezzare una lunga tabella di due pagine affinché cominci su una pagina pari o una larga tabella da mettere su due pagine affacciate con metà delle colonne nella pagina pari e l'altra metà nella pagina dispari. Gli esempi potrebbero andare avanti a lungo.

Si potrebbe pensare di usare il comando `\afterpage` fornito dal pacchetto *`afterpage`*. Ma questo comando esegue quanto indicato nel suo argomento non appena è finita la pagina corrente, indipendentemente dal numero pari o dispari della pagina appena terminata. Bisognerebbe procedere a tentoni e correggerne la posizione ad ogni modifica del testo del documento.

Abbiamo chiesto all'autore del pacchetto *`afterpage`* se non avesse voluto rimettere le mani sul suo codice per aggiungere altri comandi che verificassero

in automatico la parità del numero della pagina. Noi non volevamo mettere le mani sul suo codice, tanto più che il suo codice agiva sulla routine di uscita, cosa che solo un *TEXwizard* può essere capace di fare. David Carlisle ci ha risposto gentilmente che non se la sentiva di rimettere le mani nel suo codice scritto quindici anni prima, ma ci incoraggiava ad eseguire un comando ricorsivo di cui ci forniva il modello, dicendoci che non l'aveva controllato e amichevolmente invitandoci a tentare la sorte. Ma i veri *TEXwizard* come lui la sanno molto lunga; lo ringraziamo moltissimo per averci permesso di realizzare delle macro ricorsive a partire dalle sue indicazioni (che funzionavano benissimo, naturalmente).

Nel file `MacroGuida.sty`, che contiene le definizioni delle macro usate per comporre questo testo, appaiono le righe di codice che seguono, dove vengono definiti i due comandi `\Supaginapari` e `\Supaginadispari`, il secondo dei quali forse meno utile del primo, ma certamente non inutile; qui si illustra solo il primo.

```
\IfPackageLoadedF{afterpage}{\RequirePackage{afterpage}}

\newcommand\Supaginapari[1]{%
  \afterpage{%
    \IfOdd{\value{page}}{\Supaginapari{#1}}{#1}%
  }}
```

Poiché i due comandi si affidano all'esistenza della macro `\afterpage` definita dal pacchetto *afterpage*, per prima cosa si verifica che il pacchetto sia già stato caricato; se non lo fosse si provvede in merito. Se la chiamata fosse in seguito ripetuta, non sarebbe un problema, perché i comandi `\RequirePackage` e `\usepackage` verificano sempre da soli se il o i pacchetti indicati nel loro argomento siano o non siano già stati caricati, affinché non vengano caricati nuovamente con il pericolo di annullare eventuali ridefinizioni di comandi eseguite dal programmatore, o da altri pacchetti caricati fra la prima e la seconda chiamata.

La struttura dei due comandi è simile: per il primo comando, quando l'argomento della macro`\afterpage`, viene eseguito, è appena stata emessa una pagina e il numero di pagina è stato appena incrementato; se era stata appena emessa una pagina pari, il nuovo numero di pagina è dispari o viceversa. Quindi il test eseguito con il comando primitivo `\ifodd` verifica la disparità del contenuto del contatore `page` attraverso l'uso della macro `\value`; se questo contenuto ha la parità giusta, emette direttamente l'argomento ricevuto, altrimenti rimette in coda l'intero comando col suo argomento passandolo di nuovo al comando `\afterpage`, che lo dilaziona alla fine della pagina successiva.

Ecco quindi che i layout delle pagine affacciate di cui si parlava all'inizio di questo paragrafo è stato specificato con

```
\Supaginapari{\layout*}
```

e il disegno delle due pagine affacciate appare sempre a cominciare da una pagina pari, la prima disponibile dopo aver dato quel comando.

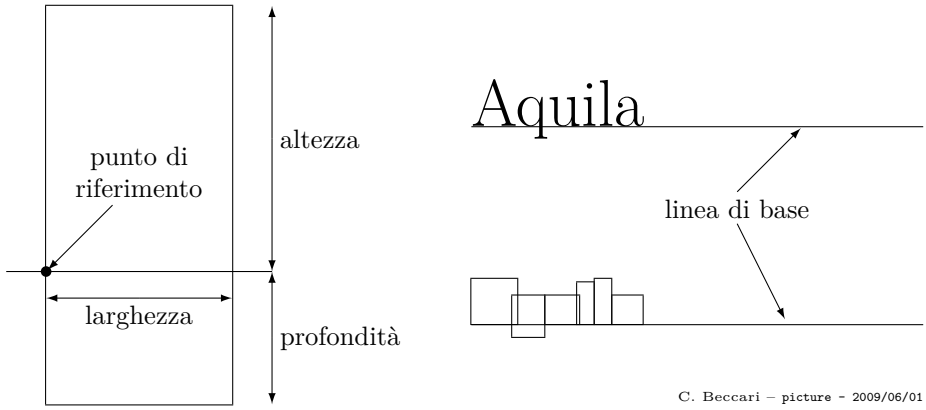


Figura 19.3: Alcune proprietà delle scatole. Nella figura a destra si vedono le scatole corrispondenti ai caratteri che formano la parola 'Aquila' allineate sulla linea di base. Le crenature fra i caratteri sono visibili dove le scatole si sovrappongono.

19.10.3 Immagini, tabelle e scatole

Il titolo di questo paragrafo è un po' enigmatico; in realtà il problema è semplice: inserire immagini nelle celle di una tabella può risultare difficile, a meno che non si sia disposti a manipolare le scatole.

Nel resto di questo testo non si è detto molto sulle scatole; l'arte di eseguire la composizione tipografica con LaTeX dovrebbe astenersi dal manipolare le scatole interne del programma di composizione, almeno in prima battuta.

Ciò non toglie che qualche nozione possa essere utile, anche per capire meglio alcuni strani costrutti eseguiti con le scatole nella definizione di alcuni comandi.

I programmi di composizione del sistema TeX maneggiano quasi esclusivamente scatole; sono scatole i singoli caratteri, le righe del testo, le pagine composte, tanto per fare esempi molto semplici. Ogni scatola è definita dal suo tipo e da tre dimensioni particolari: la larghezza, l'altezza e la profondità. Il rettangolo che può simboleggiare una scatola ha un punto di riferimento all'intersezione del bordo sinistro con la linea che separa la sua parte alta dalla sua parte profonda; il segmento intercettato su questa linea dal bordo destro e dal bordo sinistro è lungo quanto la larghezza della scatola; l'altezza è la lunghezza del segmento che parte dal punto di riferimento e raggiunge il vertice in alto a sinistra del rettangolo, mentre la profondità è la lunghezza del segmento che parte dal punto di riferimento e raggiunge il vertice in basso a sinistra; figura 19.3.

Le scatole dei caratteri vengono allineate a formare una riga semplicemente allineando i loro punti di riferimento sulla linea di base della riga composta, come si vede nella figura 19.3; il programma prende le dimensioni delle scatole che racchiudono i caratteri dal file metrico del font che sta usando; questo contiene non solo tutte le informazioni dimensionali di ciascun carattere, ma

anche le dimensioni delle crenature[18] e le possibili legature fra caratteri. La riga così formata rappresenta una scatola orizzontale, la cui larghezza è la somma delle larghezze dei caratteri componenti e degli spazi vuoti che separano le parole; l'altezza della riga è la maggiore delle altezze delle scatole allineate orizzontalmente, mentre la profondità della riga è la maggiore delle profondità delle scatole suddette; il punto di riferimento è perciò l'intersezione della linea di base con il lato sinistro del rettangolo che racchiude la riga composta.

Le scatole formate dalle righe composte vengono incolonnate in verticale allineando lungo una retta verticale i punti di riferimento di tutte le scatole orizzontali usate e intercalando alle scatole orizzontali gli spazi bianchi di interlinea e i contrografismi verticali prima e dopo ogni scatola che contenga del testo in evidenza. La pagina composta è quindi una scatola verticale; ma il programma può gestire anche altre scatole, differenti dall'intera pagina, scatole che a loro volta possono contenere altre scatole, e così via come una serie di matriosche.

LaTeX ha due tipi di scatole *orizzontali* che l'utente può usare con comandi accessibili direttamente dal compositore: `\mbox` e `\makebox`; alcune varianti sono le scatole con cornice, come `\fbox` e `\framebox`; il compositore può definire per nome dei registri-scatola nei quali può salvare scatole orizzontali mediante i comandi `\newsavebox`, `\sbox` e `\savebox`; il loro contenuto viene richiamato e usato con `\usebox`.

LaTeX ha due tipi di scatole *verticali* a cui il compositore può accedere liberamente; una è la scatola verticale prodotta con il comando `\parbox` e l'altra è la 'minipagina' ottenibile con l'ambiente *minipage*. Le scatole verticali di LaTeX possono ricevere il loro contenuto come sequenze di caratteri e di altri oggetti tipicamente di tipo orizzontale, oppure come scatole orizzontali già composte; nel primo caso LaTeX provvede a costruire le scatole orizzontali da impilare in verticale. La larghezza di simili scatole verticali coincide con quella della scatola orizzontale più larga, l'altezza e la profondità dipendono dalle opzioni che vengono specificate al comando o all'ambiente, dipendendo da come queste opzioni specificano la posizione del punto di riferimento. La scatola di tre righe Pippo può avere il suo punto di riferimento collocato in modo che la prima Pluto
Paperino

riga sia allineata con il testo circostante; oppure Pippo
Pluto
Paperino può avere l'ultima

riga allineata con il testo circostante; oppure Pippo
Pluto può avere il suo asse
Paperino

matematico allineato con l'asse matematico del testo circostante. Come si vede

[18] La crenatura (*kerning* in inglese) è l'ammontare dell'accostamento di due caratteri adiacenti, in modo da non fare apparire troppo spazio fra caratteri consecutivi dalle forme complementari; per esempio 'VA' sono accostate, mentre 'V A' non lo sono; la differenza è evidente e un font nel quale manchi l'accostamento sarebbe di qualità assai modesta.

da questo capoverso le scatole verticali non sono molto utili da usare all'interno del testo; esse trovano applicazioni particolari in altri tipi di composizione.

I comandi primitivi di TeX sono meno elaborati; i registri-scatola possono essere chiamati per nome, come in LaTeX, oppure per numero, purché questo sia strettamente compreso nell'intervallo da 0 a 254[19]; il registro-scatola 255 è riservato e l'utente non lo deve mai toccare; le scatole orizzontali sono contenute dentro `\hbox`; le scatole verticali si chiamano `\vbox`, `\vtop` e `\vcenter`; i comandi per utilizzare il contenuto delle varie scatole registrate nei registri-scatola sono invece più elaborati di quelli di LaTeX. Il lettore è invitato ad approfondire l'argomento nel TeXbook, che dedica a questi argomenti diversi capitoli.

Dopo queste premesse, possiamo tornare alle nostre immagini e alle celle delle tabelle; abbiamo visto che uno dei descrittori delle colonne è `p{`⟨*larghezza*⟩`}`; questo in sostanza definisce una scatola verticale di larghezza pari a ⟨*larghezza*⟩ dentro la quale viene composto un breve capoverso; ma questa scatola ha la *prima* riga allineata con la prima o unica riga delle celle adiacenti. Mediante l'uso del pacchetto *array* abbiamo visto che si possono usare altri due descrittori, `b{`⟨*larghezza*⟩`}` e `m{`⟨*larghezza*⟩`}` che si comportano rispettivamente come scatole verticali con l'ultima riga allineata ovvero con l'asse matematico allineato.

Le immagini introdotte nel documento ricorrendo al pacchetto *graphicx* mediante il suo comando `\includegraphics`, sono delle scatole orizzontali col loro punto di riferimento coincidente con il loro vertice inferiore sinistro. Si capisce allora che per collocare una immagine bisogna tenere conto di altezza e profondità delle scatole usate e regolarsi in modo da ottenere un buon risultato. La tabella 19.1 che si ottiene senza pensare a queste cose è orrenda. La tabella 19.2, composta ricordando le proprietà delle scatole appena descritte, invece, è composta abbastanza correttamente, anche se, come vedremo, si può fare di meglio.

La tabella 19.2 può ancora essere ritoccata aggiungendo un piccolo spazio sopra l'immagine, in modo che il filetto orizzontale non la sfiori: tabella 19.3. Questa operazione viene eseguita automaticamente se si usano il filetti orizzontali del pacchetto *bookktabs*

Infatti la tabella 19.3 è stata composta con i comandi seguenti:

```
\begin{tabular}{m{120pt}m{35mm}}
   \toprule
   Immagine & Descrizione\\
   \midrule
   \includegraphics[width=120pt]{Formaggio} &
   Il formaggio è un prodotto derivato dal latte, attraverso un
   processo che coagula la caseina, una proteina del latte.\\
```

[19] In realtà i programmi di composizione pdftex, xetex e luatex incorporano le estensioni del programma etex, come si è già spiegato nei capitoli iniziali, e quindi non è limitato nel numero dei suoi registri al numero di 255, ma ne può avere 2^{15}, numerati da 0 a $2^{15} - 1$; insomma ne può avere in pratica un numero illimitato, visto che è difficile immaginare la composizione di un documento qualsiasi che impegni simultaneamente un numero così elevato di registri-scatola.

Immagine	Descrizione
	Il formaggio è un prodotto derivato dal latte, attraverso un processo che coagula la caseina, una proteina del latte.

Tabella 19.1: Tabella in cui l'immagine e il testo adiacente hanno i punti di riferimento allineati in modo scorretto

Immagine	Descrizione
	Il formaggio è un prodotto derivato dal latte, attraverso un processo che coagula la caseina, una proteina del latte.

Tabella 19.2: Tabella in cui l'immagine e il testo adiacente hanno i punti di riferimento allineati correttamente

Immagine	Descrizione
	Il formaggio è un prodotto derivato dal latte, attraverso un processo che coagula la caseina, una proteina del latte.

Tabella 19.3: Tabella in cui l'immagine e il testo adiacente hanno i punti di riferimento allineati correttamente e dove uno spazio verticale pari a 0,8 ex separa l'immagine dal filetto sovrastante

```
   \bottomrule
\end{tabular}
```

Premesso che il descrittore di colonna `m{`⟨*larghezza*⟩`}` del pacchetto `array` risolve il problema, che cosa bisogna fare se per qualche motivo non si potesse conservare quello stesso tipo di descrizione per tutte le celle della stessa colonna? Bisogna ricorrere a comandi di basso livello, cioè ai comandi che manipolano direttamente le scatole in modo da risolvere il problema compositivo solo per la cella specifica che contiene l'immagine. Conviene anzi definirsi un comando che esegua direttamente l'inclusione dell'immagine, ma con la posizione del punto di riferimento specificata mediante una opzione. Si potrebbe definire un nuovo comando `\tabularimage` con la sintassi:

`\tabularimage[`⟨*posizione*⟩`]{`⟨*larghezza*⟩`}{`⟨*file immagine*⟩`}`

nel modo seguente:

```
\newcommand*\ttoprule{\hrule \@height0pt \@width0pt \@depth-1ex}
\newcommand*\ctoprule{\hrule \@height0pt \@width0pt \@depth.8ex}
\let\btoprule\ctoprule \let\bbotrule\ctoprule
\let\cbotrule\bbotrule \let\tbotrule\bbotrule
\newcommand*{\tabularimage}[3][b]{\parbox[#1]{#2}{%
   \csname#1toprule\endcsname %
   \includegraphics[width=#2]{#3}\par
   \csname#1botrule\endcsname}}
```

dove la ⟨*posizione*⟩ di default è la stessa che si avrebbe con il semplice comando `\includegraphics`. Con questo nuovo comando si potrebbe comporre la tabella 19.4 con i comandi:

```
\begin{tabular}{lp{35mm}}
   \hline
   Immagine & Descrizione\\
   \hline
   \tabularimage[t]{120pt}{Formaggio} &\kern-6pt
   Il formaggio è un prodotto derivato dal latte, attraverso un
   processo che coagula la caseina, una proteina del latte.\\
   \hline
\end{tabular}
```

La definizione di `\tabularimage` è un po' particolare, perché i filetti (invisibili) inseriti prima e dopo l'immagine all'interno della scatola verticale `\parbox` non possono essere i comuni filetti di LaTeX specificati con `\rule`; infatti qui abbiamo bisogno di incolonnare solo scatole verticali o altri oggetti che si possono mettere in una lista verticale; il filetto `\hrule` nonostante l'iniziale `h`, è un comando primitivo che specifica un filetto le cui altezza, larghezza e profondità sono specificate con i parametri che seguono quel comando, ma è un oggetto che può

Immagine	Descrizione
	Il formaggio è un prodotto derivato dal latte, attraverso un processo che coagula la caseina, una proteina del latte.

Tabella 19.4: Tabella con l'immagine allineata con la parte superiore delle minuscole della prima riga della seconda colonna

stare solo nelle liste di oggetti incolonnati in verticale. I punti di riferimento di questi filetti diventano quelli della scatola verticale quando le si specifichi l'opzione di ⟨*posizione*⟩ `t` oppure `b`; nel caso dell'opzione `c`, non conta tanto il punto di riferimento, quanto l'asse matematico, che TEX determina come linea mediana della scatola. Si capisce, allora, che a seconda che il collocamento della scatola corrisponda ad una delle tre specificabili, i filetti (invisibili) superiore e inferiore debbano avere caratteristiche diverse; si sono definiti sei filetti invisibili, che in realtà sono solo due distinti ma hanno sei nomi che si equivalgono; questi nomi sono ottenuti agglutinando la lettera di opzione per la collocazione con i nomi `toprule` e `botrule`; questa agglutinazione viene eseguita dalla definizione di `\tabularimage` solo al momento in cui il parametro `#1`, che corrisponde a questa opzione, è noto e l'operazione viene eseguita dalla coppia di comandi `\csname` e `\endcsname` che creano un nome di comando valido per il programma a partire da qualsiasi stringa essi racchiudano.

Questo genere di incapsulamenti di scatole con diversi punti di riferimento può essere fatto anche con le minipagine create con l'ambiente *minipage* o con le scatole orizzontali con diversi allineamenti del loro contenuto e diverse larghezze nominali.

Non si insiste oltre essenzialmente per non intimorire l'utente alle prime armi. È certo che la definizione di questi comandi, specialmente ricorrendo ai trucchi esposti, rientra nella programmazione col linguaggio LATEX, certamente non nell'arte di comporre tipograficamente con LATEX, anche se ne costituisce il supporto. Ma l'appetito vien mangiando; prima o poi tutti sentiamo il bisogno di definirci i nostri comandi, i nostri ambienti, le nostre scorciatoie per comporre ancora meglio e più correttamente.

Conclusioni

Gli ultimi esempi sono piuttosto ricchi di spunti, ma non esauriscono certamente la questione della creazione delle proprie macro. Questa è un'arte difficile, ma non

impossibile; bisogna avere la pazienza e l'umiltà di imparare senza fretta, sapendo però che la ricompensa sarà forte sotto l'aspetto della facilità del comporre.

Bisogna avere la pazienza di imparare dai libri e dai manuali esistenti, alcuni dei quali elencati nel capitolo 21. Bisogna avere anche la pazienza di leggere il codice dei file scritti da altri, in primis quelli che definiscono il formato e le classi e i pacchetti standard. Alcune parti sono veramente difficili da capire senza l'aiuto di una documentazione, ma questa è generalmente disponibile nei file `.dtx`; questi sono file TeX documentati. Se si lancia pdfLaTeX su uno qualunque di questi file, si ottiene un documento nel quale il codice è inframmezzato a spiegazioni; se queste siano utili dipende dalla competenza informatico/programmatoria del lettore, ma è certo che, superato un primo scoglio per familiarizzarsi con il linguaggio, la cosa diventa molto fruttuosa.

Capitolo 20

LaTeX: la geometria delle pagine

20.1 Introduzione

Uno degli argomenti più trattati nei forum degli utenti di TeX, non escluso il forum di GuIt, è quello dei margini, della giustezza del testo, delle testatine e dei piedini, di come fare per rendere la pagina del documento finale gradevole ed efficace.

Qualche volta si tratta di rendere un documento il più simile possibile ad una serie di altri documenti composti con altri sistemi; qualche altra volta si tratta di ricomporre un'opera antica, o comunque fuori commercio, per la quale si vuole mantenere il più possibile l'aspetto originale, e questo coinvolge anche la scelta dei font, non solo la geometria della pagina o il corpo dei caratteri.

Va allora detto prima di tutto che questo genere di attività può rendere un documento bellissimo, se il disegno della pagina e dei suoi elementi è fatto da una persona di gusto, colta, esperta di grafica editoriale o, almeno, competente a livello di 'intenditore', se non proprio di grafico professionista. Il risultato può essere pessimo se viene eseguito da una persona incompetente, ancorché animata dalle migliori intenzioni; peggio ancora se questa persona tenta di imitare lo stile di noti word processor, specialmente se i documenti scritti con quegli strumenti sono stati composti da persone che si sono divertite a scegliere corpo e stile dei caratteri senza nessun criterio in mente, spaziature, margini, inserimento di figure, creazione di tabelle, eccetera, fatte a seconda dell'ispirazione del momento e del comando più facile da raggiungere con un click di mouse.

Dietro il disegno di una pagina stampata ci sono in realtà quattro o cinque secoli di storia; considerando anche i codici manoscritti, una quindicina di secoli di storia. Questa lunghissima tradizione non può essere ignorata perché quello che la tradizione ha prodotto è quello che è sopravvissuto alla selezione naturale,

che ha tenuto conto dei mezzi a disposizione, ma anche della qualità del risultato confermata dall'apprezzamento dei lettori.

Non bisogna quindi cedere a qualunque tentazione compositiva, ma documentarsi al meglio e cercare di affinare il gusto per saper discernere i bei prodotti editoriali, dai prodotti accattivanti per colori, per font fantasiosi, per disposizione del testo che convoglia il messaggio, ma che non potrebbero essere letti da nessuno per più di due o tre pagine consecutive. Non si può confondere un opuscolo pubblicitario con un testo di poesie; l'uno e l'altro hanno i loro impieghi, ma non si possono comporre le poesie come una serie di réclame, e le réclame come dei poemi.

In parte si è trattato di questi argomenti nel capitolo 6; qui però si desidera andare più a fondo per mostrare non solo come fare, ma anche che cosa ci sta dietro; sapendo il perché, spesso si evitano errori stilistici che dovrebbero sempre essere evitati quando si cerca di comporre bene, secondo l'arte tipografica.

È chiaro a tutti che la parte più importante di un documento è il suo contenuto. Questo aspetto non verrà trattato in questa guida; esistono ottime grammatiche e testi di 'retorica' per scrivere in modo efficace e che sappia convogliare il messaggio giusto al lettore.

Uno degli aspetti più importanti in questa azione è proprio quello di eseguire una buona analisi del lettore; se si riesce a fare un buon profilo del lettore, allora è possibile confezionare il messaggio in modo che sia adatto a quel tipo di lettore. È facile capire che un testo destinato a una moltitudine di lettori dai profili differenti può essere difficilissimo da scrivere; se si scrive per i lettori più informati, quelli meno informati non capiscono; se si scrive per i lettori meno informati, gli altri si annoiano. Questo stesso testo di introduzione all'arte tipografica con LaTeX, che si rivolge ad una moltitudine di lettori, cerca di essere un compromesso fra i lettori esperti e quelli neofiti; sicuramente non è completamente adatto né ai primi né ai secondi, tuttavia si affida alla loro comprensione nella speranza che i primi possano trovare qualche spunto che non conoscevano e che i secondi siano invogliati ad andare più a fondo.

Certamente LaTeX consente di dedicarsi con tutta l'attenzione possibile al confezionamento del messaggio; alla sua corretta scrittura, alla strutturazione, all'evidenza da dare ai concetti base, eccetera. Tuttavia questi aspetti vengono sottolineati meglio da una particolare disposizione del testo sulla pagina, dai font usati, dalle decorazioni, dai colori, eccetera. Questi aspetti fanno parte della grafica editoriale; il grafico che si accinge a definire un layout particolare deve stare attento a non esagerare perché il layout non deve attrarre o distrarre l'attenzione del lettore, specialmente in un documento contenente un testo molto strutturato come può essere un libro. Il layout deve essere sobrio, aiutare il lettore nella sua lettura, evidenziare ciò che deve essere evidenziato, senza richiamare su di sé l'attenzione del lettore. Aprendo un libro qualsiasi ben disegnato si ha una immediata percezione dell'equilibrio grafico anche senza leggerne il testo; si percepisce una sensazione confortevole che invoglia a leggere. Un libro mal disegnato attrae l'occhio sui fronzoli, le decorazioni, i colori, ma distrae dalla lettura.

Vale la pena di riportare il commento di Peter Wilson, l'autore della classe *memoir*, a questo proposito[1]:

> L'essenza della buona tipografia è che essa non viene notata alla prima occhiata, forse nemmeno alla seconda o alle successive, se non da chi abbia un occhio esercitato. Se la vostra reazione iniziale quando guardate un libro è quella di esclamare qualcosa a proposito del suo layout, allora probabilmente quel libro è mal progettato, ammesso che sia mai stato progettato. La buona tipografia è fatta di sottigliezze, non di esagerazioni.

Qui di seguito si indicheranno alcuni aspetti della grafica editoriale che derivano dalla tradizione e si cercherà di spiegare da dove derivano alcune di queste 'norme' (piuttosto elastiche, per fortuna); si indicheranno anche alcuni pacchetti di estensione che consentono di personalizzare diverse parti del layout.

20.2 La geometria della pagina

Il lettore non si preoccupi se finora si è usata la parola 'layout' senza definirla. In inglese la parola deriva dalla locuzione verbale *to lay out* che vuol dire adagiare, distendere; ma in tipografia essa non implica solo la disposizione geometrica delle varie parti che compongono una pagina, ma anche i font e le loro dimensioni, gli spazi (i contrografismi) fra le varie parti del testo, e tanti altri dettagli più o meno evidenti, che non hanno a che fare solo con la geometria della pagina, ma hanno a che fare con la pagina nel suo complesso.

Bisogna innanzi tutto stabilire di che tipo di testo si vuole confezionare il disegno: è un testo di consultazione, un testo letterario, un romanzo, un saggio prevalentemente testuale, un testo tecnico-scientifico, un testo artistico con molte immagini; è un testo di lusso o si tratta di dispense; richiede una lettura continua, oppure può o deve essere letto saltando da una parte all'altra; deve essere stampato in grande formato o in formato tascabile.

Come si vede, già queste poche cose richiedono diverse scelte grafiche, alcune delle quali, talvolta, sono più di tipo economico che di tipo estetico.

20.2.1 Il formato delle pagine

In Italia un formato corrente per libri di ogni genere rilegati in brossura è quello la cui pagina rifilata ha le dimensioni di 170 mm per 240 mm; misurando alcuni libri presi a caso, chi scrive ha trovato dimensioni leggermente diverse, di solito non più del 10% dai valori indicati. Libri più importanti, rilegati con copertina rigida (incassati) possono avere dimensioni maggiori, anche molto maggiori. I

[1] The essence of good typography is that it is not noticeable at first, or even second or later, glances to any without a trained eye. If your initial reaction when glancing through a book is to exclaim about its layout then it is most probably badly designed, if it was designed at all. Good typography is subtle, not strident.

Pieghe	Nome	Formato del foglio da piegare					Pagine	Facciate
		A0	A1	A2	A3	A4		
1	folio	A1	A2	A3	A4	A5	2	4
2	quarto	A2	A3	A4	A5		4	8
3	ottavo	A3	A4	A5			8	16
4	sedicesimo	A4	A5				16	32
5	trentaduesimo	A5					32	64

Tabella 20.1: Formati delle pagine a seconda delle dimensioni del foglio di partenza e del numero di pieghe

libri di consultazione, anche se incassati, possono avere dimensioni minori, con la pagina in formato A5, cioè di 148 mm per 210 mm. Dimensioni minori, per esempio A6 di 105 mm per 148 mm, si incontrano in opuscoletti decisamente tascabili e di piccolo spessore.

La scelta dei formati dipende dalla carta disponibile e dal numero di pieghe che si possono fare per creare ogni segnatura. Oggi si tende a stampare a nastro continuo, mentre in passato si stampava su grandi fogli di dimensioni standardizzate. Il nastro continuo permette di tagliare delle sezioni più o meno lunghe che svolgono la funzione dei grandi fogli isolati del passato, ed è per questo che con il nastro continuo è facile avere più libertà nel formato della pagina. Tuttavia, una volta piegato il foglio un certo numero di volte, si ottiene una 'segnatura', un fascicoletto contenente un numero di facciate pari ad una potenza di due, se le pieghe sono sempre state fatte 'a metà'; talvolta si fanno delle pieghe in tre, per cui il numero di facciate di una segnatura può anche essere un multiplo di sei.

Un grande foglio piegato solamente in due produce una segnatura con due pagine e quattro facciate di solito di grandi dimensioni; in passato era comune, per esempio, per i messali; nelle nostre università, prima di essere sostituiti dalle registrazioni informatiche, i registri relativi agli studenti erano spesso dei libroni 'in folio' con le pagine di circa 400 mm per 600 mm.

Con un'altra piega a metà si hanno segnature con quattro pagine e otto facciate; sono segnature in 'quarto'. Con un'altra piega a metà si hanno segnature con otto pagine e sedici facciate; sono segnature in 'ottavo'. Via via che si piega a metà l'ordinale che indica la segnatura, sedicesimo, trentaduesimo, ricorda sempre una potenza di due; si capisce però che più si piega, più diventa spessa la piega dove bisogna cucire e meno precisa diventa la posizione della parte stampata su ogni facciata. La tabella 20.1 si riferisce alle piegature dei fogli della serie A ISO-UNI; il nome della piegatura, folio (o quartino), quarto, eccetera, dà luogo a formati sempre della serie A ma più piccoli. Evidentemente al crescere del numero di pieghe, le pagine e le facciate diventano sempre più piccole e sempre più numerose; la tabella in questione si riferisce comunque sempre a piegature a metà, non anche in terzi, per cui le successive piegature danno sempre luogo a

formati della serie A.

Eseguendo delle pieghe in tre si possono avere delle segnature in 'sesto', in 'dodicesimo o in 'ventiquattresimo, ma sono segnature meno frequenti e che richiedono fogli iniziali di dimensioni insolite.

A prescindere dal numero di pagine che formano ogni segnatura, questa rappresenta il risultato della piegatura dei grandi fogli e anche la 'materia prima' per la cucitura a macchina o a mano per confezionare il libro; quando questo è cucito, prima o dopo venire brossurato, sicuramente prima di venire incassato, il blocco delle segnature cucite viene rifilato su tre lati per avere le pagine tutte delle stesse dimensioni e per eliminare quei piccoli difetti di allineamento che potrebbero essersi formati durante la cucitura. Il libro brossurato può avere la copertina un poco più ampia delle pagine che racchiude; il libro incassato ha sempre la copertina che sporge oltre le pagine; il libro incassato spesso ha una sovracoperta, destinata a proteggere la copertina rigida che talvolta è realizzata con coperture di materiali pregiati come stoffa o pelle.

20.2.2 Le segnature e le imposizioni

Questa rapida scorsa sulla fabbricazione del libro permette di capire che il 'desk top publishing', a cui spesso ci si riferisce quando si usano sistemi di composizione 'fai da te', non può prescindere dalla conoscenza di queste regole elementari, anche quando il testo composto su fogli A4, eventualmente 'in folio' e perciò con pagine in formato A5, sono incollati a caldo e brossurati con metodi economici; anche in questi casi è necessario un passaggio alla taglierina per rendere i tagli superiore, inferiore ed esterno ben pareggiati e facili da far scorrere fra le dita. LaTeX può essere utile in questi casi; sicuramente è molto utile per il lavoro tipografico a foglio continuo, anche se il suo file di uscita contiene una sequenza di pagine non necessariamente nella sequenza giusta e ruotate quanto basta per comporre i due lati dei grandi fogli che formeranno ciascuna segnatura; questa operazione verrà eseguita in tipografia con software adeguato e con macchine adatte per marcare le grandi lastre di fotolitografia con le quali verrà eseguita la stampa con foglio continuo.

A casa, per produrre un fascicoletto o un libro in formato A5, il sistema TeX offre un certo numero di possibilità; dal punto di vista delle macchine bisogna immaginare di disporre di una stampante che possa stampare fronte e retro; in mancanza bisogna stare attenti a stampare prima tutti i fronti; poi ricaricare a rovescio il pacco dei fogli stampati da un lato per stamparli dall'altro; è ovvio che questa operazione richiede la massima attenzione per essere sicuri che ogni foglio risulti stampato correttamente sui due lati con quello che deve contenere. Meglio ancora sarebbe disporre di una stampante che accetti in entrata fogli in formato B4 e che possa stampare fronte e retro, con le attenzioni esposte sopra nel caso essa possa stampare solo fronte. Il vantaggio sarebbe che piegando a metà segnature su fogli B4, si ottengono pagine in formato B5, cioè di 176 mm per 250 mm, che ammettono abbastanza margine, ma senza eccessivo sfrido, per

stampare pagine in formato 170 mm per 240 mm, il formato della maggior parte dei libri in Italia.

Ma per stampare questi fogli è necessario riordinare le pagine; bisogna decidere innanzi tutto se si vogliono fare segnature di quattro fogli (otto pagine e sedici facciate) o segnature con più o meno fogli. Bisogna comporre il file in formato PDF, magari correggendo i margini affinché le pagine appaiano alla giusta distanza dalla piega centrale di ogni foglio A4[2]. Poi bisogna eseguire l'*imposizione*, cioè bisogna riordinare e ruotare le pagine affinché appaiano nell'ordine e nel verso giusto quando si compone la segnatura; per fare questo si invoca il pacchetto *pdfpages*: per esempio, con il file PDF che si chiami `libretto.pdf`, che contenga pagine in formato A5, e disponendo di una stampante A4, bisogna predisporre un piccolo file `.tex` così composto:

```
% File segnature.tex
\documentclass{article}
\usepackage[final]{pdfpages}
\begin{document}
\includepdf[pages=-,nup=1x2,landscape,signature=16]{libretto.pdf}
\end{document}
```

Compilando questo file tramite pdflatex si ottiene un file con le pagine A4 in posizione 'panoramica' (*landscape*), su ogni facciata della quale si trovano due pagine A5 estratte dal file iniziale, e ordinate correttamente per formare segnature di sedici facciate; quindi la prima segnatura avrà un primo foglio che sul fronte avrà le pagine 16 e 1 e sul retro le pagine 2 e 15; un secondo foglio con le pagine 14 e 3 sul fronte e 4 e 13 sul retro; un terzo foglio con le pagine 12 e 5 sul fronte e 6 e 11 sul retro; un quarto foglio con le pagine 10 e 7 sul fronte e 8 e 9 sul retro. Secondo questa stessa sequenza vengono imposte le ulteriori pagine sui fogli che formeranno le segnature successive.

Stampando il file `segnature.pdf` sulla stampante A4 in modalità fronte e retro, i vari fogli vengono stampati in ordine; ogni quaterna di fogli contiene già una segnatura che basta piegare a metà per poi impilarla con le successive segnature in bell'ordine; non resta che incollare o cucire e poi brossurare o incassare. Per maggiori dettagli ci si può riferire alla documentazione del pacchetto *pdfpages*.

Esiste anche il pacchetto *booklet* per ottenere più o meno lo stesso risultato; le differenze sostanziali fra le strategie di imposizione dei due pacchetti sono essenzialmente queste:

- *pdfpages* presuppone che il documento di cui si devono creare le imposizioni sia già composto in formato PDF; *booklet* invece compone il documento a partire dal solito file sorgente producendo in uscita un file PDF contenente le imposizioni. Questo implica, specialmente con le segnature con più

[2]Si suppone che a casa si disponga dei una normale stampate per fogli A4; se si disponesse di una stampante per fogli A3, quanto viene detto qui vale per un fascicolo in formato A4, oppure per *segnature* in sedicesimo (otto fogli, sedici pagine, 32 facciate) in formato A5.

pagine, che si metta a dura prova l'intera memoria del programma di composizione; questa possibilità è menzionata anche nella documentazione di `booklet`.

- Il testo da comporre potrebbe non avere un numero di pagine complessivo pari ad un multiplo intero del numero di pagine delle segnature specificate; entrambi i pacchetti aggiungono il numero necessario di pagine bianche all'ultima segnatura. Ma `pdfpages` aggiunge un numero di pagine tale che l'ultima segnatura abbia lo stesso numero di pagine delle altre, mentre `booklet` consente che l'ultima segnatura abbia un numero di pagine multiplo di 4, anche se questa segnatura dovesse avere meno pagine delle altre.

Il testo di Gustavo Cevolani [15] spiega molto dettagliatamente il procedimento delle imposizioni con molti esempi ed esegue un confronto accurato dei due pacchetti.

Naturalmente per un lavoro non casalingo basta predisporre il file PDF, magari con i *crocini* (vedi più avanti) e in tipografia verranno eseguite le imposizioni corrette per la produzione delle segnature sui grandi fogli da piegare con diverse pieghe, per poi passare le segnature alla sezione di cucitura o incollatura e poi alla sezione di confezionamento finale con la coperta in brossura o incassata, prodotte da una linea di produzione diversa.

Con riferimento alla figura 20.1 vale la pena di osservare come sono eseguite le imposizioni per generare le segnature su un foglio piegato a metà, piegato due volte a metà e piegato tre volte a metà, oppure, per la segnatura in sesto, piegato in tre e poi a metà. Le pieghe si suppongono sempre eseguite piegando a metà (o in tre parti) il lato lungo del foglio o del fascicoletto parziale; le pieghe disegnate in rosso rappresentano quella che diventerà il dorso della segnatura, la piega sulla quale viene eseguita la cucitura o la spillatura; le pieghe disegnate in nero saranno eliminate con la rifilatura dei tre bordi esterni del fascicolo o del libro.

20.2.3 I crocini

Per eseguire il lavoro in tipografia è indispensabile poter disporre di una estensione a LaTeX che permetta di segnare i 'crocini' in italiano, i 'crop marks' in inglese; si tratta di segni spesso a forma di croce che permettono di registrare con precisione le varie pagine durante il lavoro fatto in tipografia, in modo che le stampe delle due facciate di una stessa pagina siano esattamente allineate: guardando in controluce, il margine sinistro del testo della facciata sul verso deve corrispondere esattamente con il margine destro del testo stampato sul recto della stessa pagina e lo stesso deve succedere per tutti gli altri margini corrispondenti.

Ogni compositore può crearsi i suoi crocini ricorrendo all'ambiente *picture* come nella figura 20.2, che mostra un esempio predisposto per una tipografia italiana; in essa appaiono sia i mezzi crocini che indicano i limiti della pagina rifinita e i crocini interi che servono per registrare con precisione le imposizioni. Non è difficile fare una cosa come quella mostrata nella figura 20.2, ma è delicata;

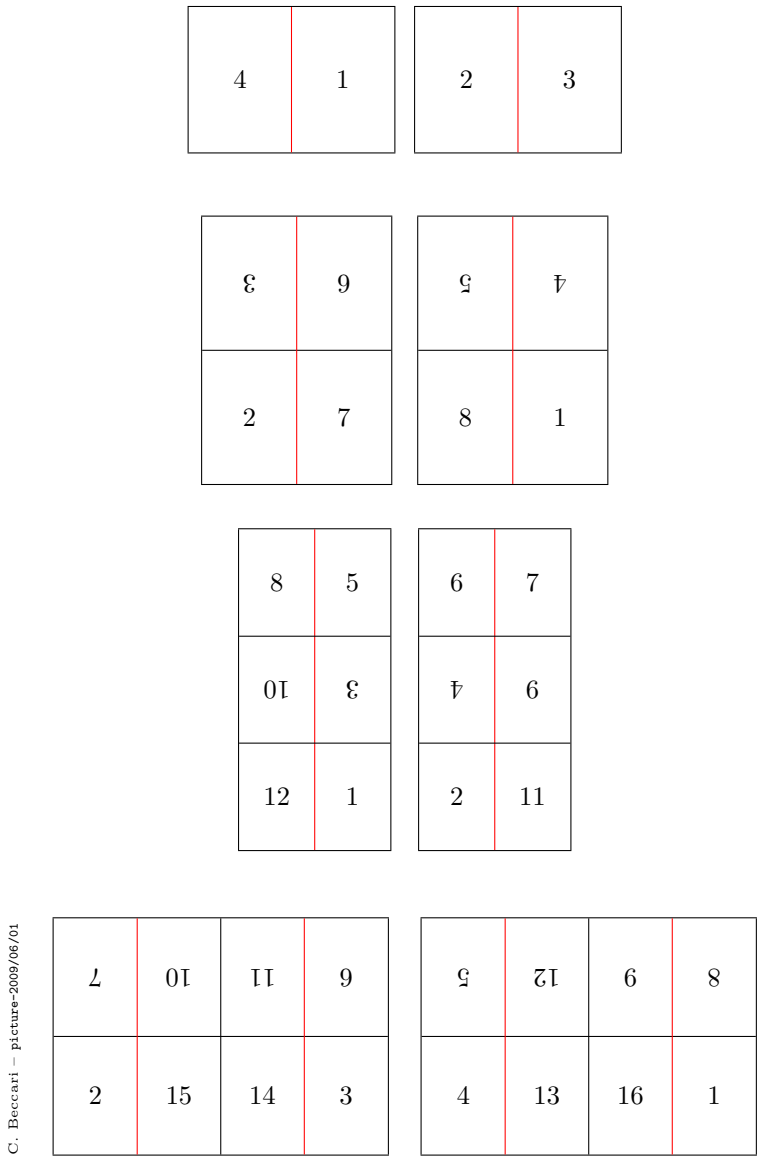


Figura 20.1: Le imposizioni per generare una segnatura in folio, in quarto, in sesto e in ottavo

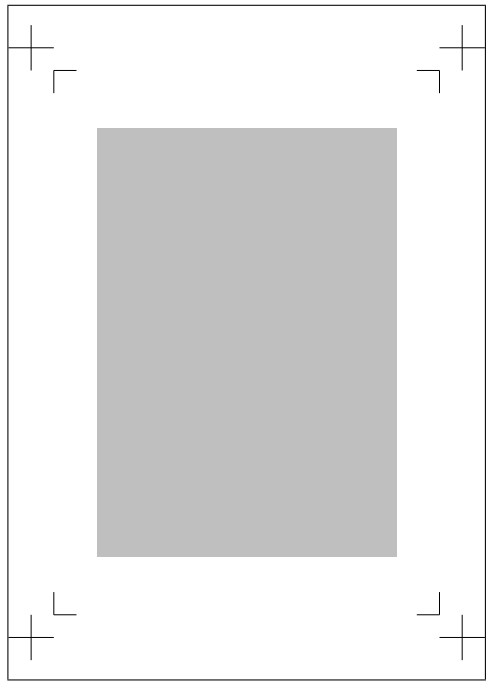


Figura 20.2: Esempio di crocini predisposti per una tipografia italiana

in alternativa si può ricorrere al pacchetto *crop* che provvede da solo a disporre i crocini (preconfezionati secondo la tradizione USA) nelle posizioni specificate in relazione alle dimensioni della pagina rifinita. La documentazione si trova in `.../doc/latex/crop/crop.pdf`.

20.2.4 Dimensioni della gabbia del testo

Su ogni facciata va collocata *la gabbia del testo* stampato, chiamata anche *specchio di stampa* o *gabbia interna*; questa è la parte più significativa della pagina e va dimensionata con cura. Non è solo una questione di margini: il rettangolo che contiene il testo deve essere adeguato al testo che esso contiene e ai font con cui è scritto.

Studi di carattere psicosociologico hanno messo in evidenza che il lettore medio si trova a suo agio nel leggere se le righe non contengono più di 65–70 caratteri, spazi e segni di interpunzione compresi (regola di Bringhurst). La scelta del font e la scelta del corpo normale, quello con cui è scritto il testo corrente, diventano perciò fondamentali per definire la giustezza, cioè la base del rettangolo che circoscrive il testo.

Esistono tabelle per determinare la giustezza ottimale per ciascun font e per ogni corpo dello stesso font; tuttavia non è facile procurarsi queste tabelle per

qualunque font. LaTeX però offre un metodo semplice e pratico per determinare la giustezza migliore con un dato font. Basta predisporre un piccolo file di prova nel quale si sceglie il font e con quello si misura la lunghezza dell'alfabeto minuscolo.

Per esempio, con il carattere Latin Modern, il semplice file seguente permette di sapere che la larghezza dell'alfabeto minuscolo di corpo 10 pt vale 137,58371 pt:

```
\documenctlass{minimal}
\newlength{\alfabeto}
\begin{document}
\settowidth{\alfabeto}{abcdefghijklmnopqrstuvwxyz}
L'alfabeto minuscolo LM a 10\,pt vale \the\alfabeto.
\end{document}
```

Sapendo che ci sono 72,27 punti in un pollice e che un pollice vale 25,4 mm, si può determinare una volta per tutte la dimensione in millimetri di un punto; precisamente la costante di conversione vale 0,351 459 804 mm/pt. Con questa costante di conversione possiamo quindi determinare che l'alfabeto minuscolo del font Latin Modern di 26 lettere è lungo 48,36 mm, quindi si ha mediamente 1,859 813 mm/lettera. Questo semplice conto svolto una volta per tutte permette di calcolare che con questo font la giustezza ottimale di 65–70 caratteri per ogni riga vuol dire una giustezza da 120 mm a 130 mm.

Esercizio 20.1 Questo testo è stato composto con il carattere Latin Modern a 10 punti con la classe *book* senza modifiche sostanziali. Il file `bk10.clo`, che specifica le misure relative al font di 10 pt, dice che la giustezza `\textwidth`, quando si compone ad una colonna, è impostata a 345 pt; la giustezza è nell'intervallo indicato sopra?

Il conto appena fatto potrebbe sembrare troppo semplificato; non si è tenuto conto degli spazi e dei segni di interpunzione; siccome gli spazi interparola sono dello stesso ordine di grandezza di una 'e' (anche se possono risultare un poco allargati o ristretti per la giustificazione) e i segni di interpunzione più frequenti (il punto e la virgola) contano poco, anche se contano gli spazi da cui sono seguiti, il conto appena fatto, benché approssimato, fornisce una eccellente stima della lunghezza della riga ideale.

A titolo di esempio nella tabella 20.2 sono riportate le giustezze in millimetri per righe composte con 60, 65, 70 e 75 caratteri con diversi font. I dati relativi ai font CM e LM sono praticamente identici, mentre si osservano diversità sensibili nelle giustezze ottimali con i font TX e PX. Ma una differenza che non emerge a colpo d'occhio è la seguente: rispetto alle giustezze relative al corpo di 10 pt, le giustezze con corpi maggiori o minori per i font TX e PX sono strettamente proporzionali al corpo, perché con questi font LaTeX lavora solamente con il corpo di 10 pt ingrandito o rimpicciolito a seconda della necessità. Con i font CM ed LM cade la proporzionalità, perché LaTeX con questi font usa caratteri disegnati apposta per ciascun corpo[3]. Il corpo di 10 pt è il corpo del testo corrente con

[3]Veramente il corpo indicato con 11 pt in realtà è di 10,95 pt e viene ottenuto per ingrandimento del font CM a 10 pt, e per rimpicciolimento del font LM a 12 pt.

Giustezze in millimetri				
	Numero di caratteri			
Corpo	60	65	70	75
8	87	95	102	109
9	95	103	111	119
10	103	112	120	129
11	113	122	132	141
12	121	131	141	151

(*a*) Font CM

Giustezze in millimetri				
	Numero di caratteri			
Corpo	60	65	70	75
8	87	95	102	109
9	95	103	111	119
10	103	111	120	129
11	111	120	129	138
12	121	131	141	151

(*b*) Font LM

Giustezze in millimetri				
	Numero di caratteri			
Corpo	60	65	70	75
8	77	83	90	96
9	87	94	101	108
10	96	104	113	121
11	106	115	124	133
12	116	125	135	145

(*c*) Font TX

Giustezze in millimetri				
	Numero di caratteri			
Corpo	60	65	70	75
8	86	93	100	108
9	97	105	113	121
10	108	117	126	135
11	118	128	138	148
12	129	140	151	162

(*d*) Font PX

Tabella 20.2: Giustezze determinate con il metodo della lunghezza dell'alfabeto. Le giustezze sono espresse in millimetri e i valori sono arrotondati al valore intero. (*a*) font CM (Computer Modern); (*b*) font LM (Latin Modern); (*c*) font TX (Times eXtended); (*d*) font PX (Palatino eXtended).

l'opzione *10pt*, il corpo 11 pt è relativo all'opzione *11pt* e, corrispondentemente, il corpo di 12 pt si riferisce all'opzione *12pt*; per il corpo di 10 pt il corpo di 9 pt si riferisce al comando `\small` e quello di 8 pt al comando `\footnotesize`. Sotto questo aspetto i font nativi del sistema TEX sono migliori di quelli commerciali distribuiti nel solo corpo a 10 pt. Fra i font commerciali esistono anche font 'esperti' che hanno forme diverse a seconda del corpo, ma sono meno diffusi.

Il font usato determina pertanto un intervallo ottimale entro cui scegliere il valore della giustezza. Come si determina l'altezza? La risposta dipende molto dalla natura della testatina e del piedino.

20.2.4.1 Testatine e piedini; il pacchetto *fancyhdr*

Bisogna innanzi tutto decidere che cosa contengano le testatine e i piedini delle pagine dispari e di quelle pari. Se il testo è un romanzo, le testatine e i piedini servono a poco e possono essere anche vuoti o uno dei due può contenere solo il numero della pagina; quest'ultimo, se è l'unica indicazione oltre al testo presente sulla pagina potrebbe essere collocato nel piedino, centrato o a filo del margine

esterno del blocco di testo, oppure potrebbe essere collocato in testa o al piede del margine esterno della pagina (come una nota marginale). In entrambi i casi esso non impegna visivamente la pagina, quindi l'altezza del blocco di testo contiene effettivamente solo il testo.

Se invece il testo è un documento di consultazione, le testatine destra e sinistra, e i piedini destro e sinistro giocano un ruolo essenziale nella 'navigazione' del testo alla ricerca delle informazioni che interessano. Ecco allora che il pacchetto *fancyhdr* diventa utilissimo per costruire testatine e piedini fortemente strutturati che aiutino davvero nella navigazione del testo; questo pacchetto consente di gestire l'informazione variabile di pagina in pagina da collocare nelle testatine di sinistra e di destra; lo stesso vale per i piedini. Il pacchetto consente di immaginare ogni testatina o piedino diviso in tre parti, quella di sinistra, quella di centro e quella di destra e di scrivere in ciascuna parte una informazione diversa. La documentazione del pacchetto *fancyhdr* si trova in `.../doc/latex/fancyhdr/fancyhdr.pdf`.

Si capisce subito che questo genere di testatine e di piedini fortemente strutturati possono essere molto utili nei rapporti tecnico-scientifici; adeguatamente predisposti possono indicare in un vocabolario il primo e l'ultimo lemma che compaiono in una pagina. Ma quello che è più importante è che questi elementi di riferimento sono otticamente 'ingombranti' e quindi bisogna ritagliarli dal rettangolo del testo, tenendo conto anche di un adeguato spazio di separazione, eventualmente anche di un filetto di delimitazione.

20.2.4.2 Le proporzioni della gabbia di testo

Stabilito se le testatine e/o i piedini facciano o non facciano parte della gabbia del testo, bisogna decidere quali proporzioni abbia la gabbia del testo, il rettangolo che racchiude il testo, in relazione al rettangolo che rappresenta la pagina rifinita. Si può decidere che abbia le stesse proporzioni (lo stesso rapporto altezza/base) come si può optare per proporzioni diverse. Ma non si può prendere questa decisione se non si prendono in considerazione anche i margini e ciò che eventualmente questi devono contenere.

Una decisione preliminare molto importante è se si stampa solo sul recto o sul recto e il verso (in bianca e volta, come si dice in gergo tipografico). In realtà se si stampa solo in bianca, lo stile grafico perde abbastanza la sua importanza, perché si sta parlando di un fascicolo o incollato a caldo o semplicemente inserito in una legatura a spirale, o a punti o ad anelli. Ciò non toglie che non si debba prendere qualche precauzione per evitare che una parte di testo risulti illeggibile.

Normalmente se si stampa solo in bianca le proporzioni della gabbia del testo rispecchiano le proporzioni della pagina fisica, eventualmente depurata dell'ulteriore margine a sinistra per tener conto della legatura artigianale o dei fori per gli anelli; su un foglio A4 si può pensare di assegnare circa 7-8 mm per la rigidezza della incollatura a caldo o della legatura a spirale, un poco di più per i fori per i raccoglitori ad anelli. La larghezza utile della pagina diventa perciò da

202-203 mm a 195 mm e la proporzione altezza su larghezza va da circa 1,47 a circa 1,52, invece del valore di $1.41 \approx \sqrt{2}$ tipico della serie UNI.

Per esempio, per un raccoglitore ad anelli il rapporto utile vale 1,52. Se la giustezza è di 125 mm, allora l'altezza della gabbia del testo deve essere di 190 mm. Centrando la pagina nello spazio utile, i margini superiore e inferiore diventano di $(297\,\text{mm} - 190\,\text{mm})/2 = 53{,}5\,\text{mm}$. I margini destro e sinistro diventano invece diversi se si deve tenere conto dei 15 mm assegnati allo spazio per i fori degli anelli; il margine destro diventa $(210\,\text{mm} - 15\,\text{mm} - 125\,\text{mm})/2 = 35\,\text{mm}$ e il margine sinistro è 15 mm più grande e risulta 50 mm; come controprova: $50\,\text{mm} + 125\,\text{mm} + 35\,\text{mm} = 210\,\text{mm}$.

Per la stampa in bianca e volta c'è maggiore libertà, sia pure tenendo conto che le pagine destre e sinistre sono diverse, se non altro perché i margini interni devono essere diversi dai margini esterni; anche in questo caso si può tenere conto di qualche millimetro per contrastare l'effetto di curvatura in corrispondenza del margine interno, il margine di cucitura. Tuttavia non si è tenuti a rispettare la stessa proporzione della pagina fisica. Anzi, i grandi book designer giocano con la forma della pagina e la forma della gabbia del testo per ottenere effetti speciali. Bringhurst, per esempio, gioca con i rapporti geometrici come se fossero i rapporti di note musicali sia della scala diatonica sia di quella cromatica e ricerca i rapporti che formino accordi gradevoli; altri invocano il rapporto aureo; altri invocano i rapporti che esistono negli elementi delle figure geometriche, in particolare dei poligoni regolari. Un esempio è dato dal disegno riportato nel capitolo 6 nella figura 6.2. Definita la diagonale di un poligono regolare come la corda più lunga, allora la diagonale di un triangolo equilatero sta al lato in rapporto unitario e dà quindi luogo ad una gabbia quadrata. Con il quadrato il rapporto è $\sqrt{2}$; con il pentagono il rapporto è quello aureo 1,618 . . . ; con l'esagono il rapporto vale 2, con l'ettagono il rapporto vale 2,247; con l'ottagono il rapporto vale 2,613. Raramente si usano rapporti così grandi (superiori a 2) ma non sono esclusi per particolari documenti; in un catalogo di una casa editrice c'era una gabbia con il rapporto pari a 2,25, praticamente il rapporto dell'ettagono.

20.2.4.3 I margini

Tradizionalmente i margini inferiore e superiore stanno fra di loro come il margine esterno sta al margine interno. Inoltre in una coppia di pagine affacciate (in inglese uno *spread*) spesso si desidera che la coppia di margini interni siano complessivamente equivalenti a ciascuno dei margini esterni. Esaminando il layout di diversi testi antichi e moderni, Wilson ha trovato che in realtà questa tradizione non era rispettata nemmeno secoli fa, sebbene moltissimi libri pubblicati in passato la rispettassero. Wilson ha raccolto diversi disegni di layout antichi e moderni nella breve documentazione storica che si può leggere con `texdoc memdesign`.

Per seguire questa regola gli antichi usavano squadra e compasso ed eseguivano delle ingegnose costruzioni geometriche per determinare le proporzioni della gabbia e dei margini.

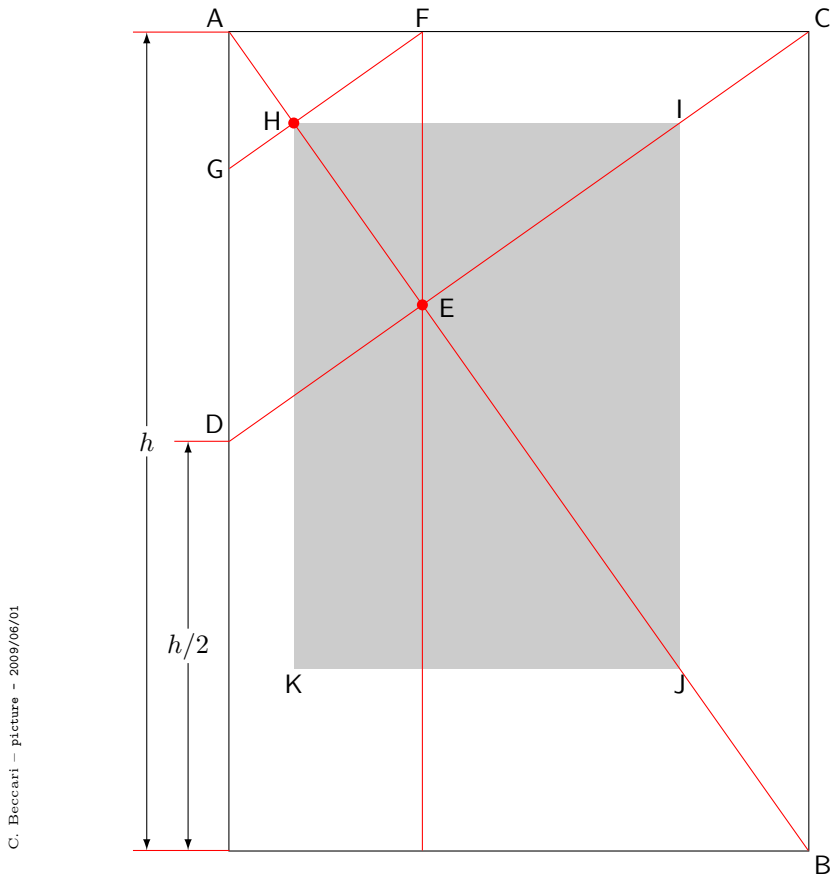


Figura 20.3: Costruzione geometrica usata da Gutenberg per determinare le proporzioni e la posizione della gabbia delle sue famose Bibbie

Gutenberg avrebbe disegnato la gabbia delle sue famose Bibbie e di diversi altri stampati seguendo la costruzione della figura 20.3. La costruzione è più lunga da commentare che non da valutare graficamente. Comunque si basa sul metodo delle due diagonali. La diagonale AB indicata nella figura è la diagonale della pagina di destra, mentre la diagonale CD è la diagonale delle due pagine affacciate, ed è per questo che interseca il margine sinistro della pagina di destra a metà altezza. Le due diagonali si intersecano nel punto E per il quale si traccia la verticale EF; di qui si traccia la parallela FG alla diagonale delle due pagine affacciate; il punto di intersezione H di questa parallela con la diagonale della pagina individua il primo vertice della gabbia dal quale è semplice tirare le debite linee verticali ed orizzontali per completare il rettangolo HIJK della gabbia.

Analizzando la costruzione geometrica si determina abbastanza facilmente che il risultato della costruzione di Gutenberg è lo stesso di quello che si ottiene con il metodo delle nove strisce descritto qui di seguito. Infatti la prima metà

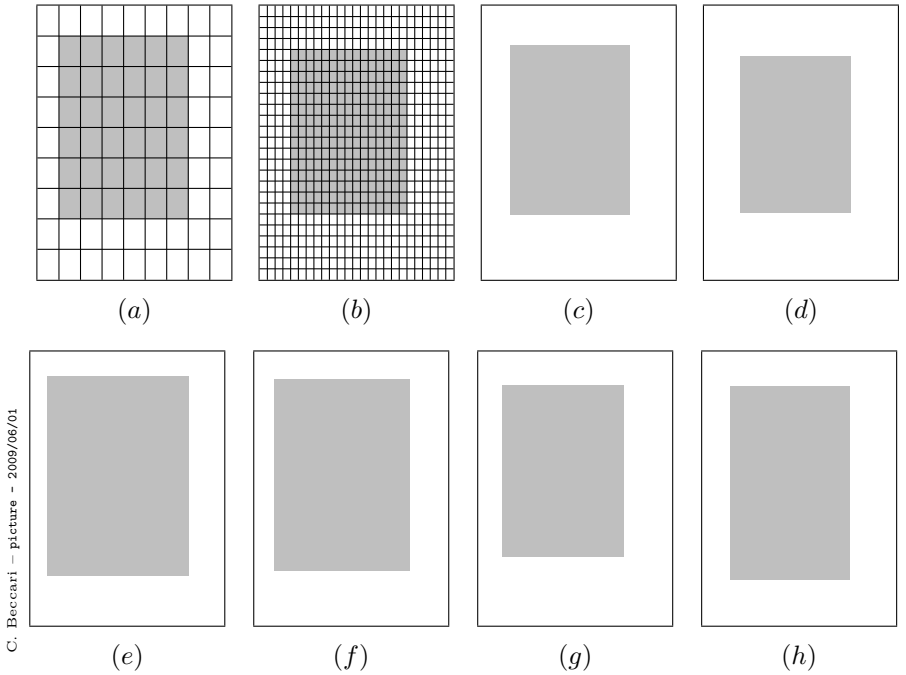


Figura 20.4: Determinazione della gabbia e dei margini con il metodo delle strisce. (*a*) esempio con 9 strisce e rapporto 2 : 1; (*b*) esempio con 25 strisce e rapporto 3 : 2; (*c*) esempio con il rapporto aureo – si noti che la gabbia non è un rettangolo aureo; (*d*) esempio con il rapporto 4 : 3. Nella seconda riga viene rappresentato il metodo delle strisce riferito solo al rapporto dei margini: (*e*) metodo dei margini con 11 strisce; (*f*) metodo delle 10 strisce usato dalla classe `octavo`; (*g*) metodo dei margini con 8 strisce; (*h*) gabbia aurea con margini aurei su foglio A4

della costruzione determina il punto E tale che il segmento AE è un terzo del segmento AB; la seconda parte della costruzione determina il punto H in modo che nuovamente il segmento AH sia un terzo del segmento AE; perciò il margine interno e quello superiore sono rispettivamente un nono della larghezza e dell'altezza della pagina rifinita sulla base della quale si è fatta la costruzione geometrica. Per la similitudine dei vari triangoli che hanno generato il margine interno e quello superiore, i margini esterno e quello inferiore sono esattamente il doppio, quindi due noni della corrispondente dimensione della pagina rifinita.

La costruzione più semplice consiste nel dividere il rettangolo della pagina rifinita un un certo numero di strisce uguali sia in verticale sia in orizzontale; due strisce verticali esterne vengono riservate al margine esterno e due strisce orizzontali in basso vengono riservate al margine inferiore; una striscia orizzontale in alto viene riservata al margine superiore e, infine, una striscia verticale viene riservata al margine interno. In questa maniera le proporzioni della gabbia del testo risultano uguali a quelle della pagina; il margine interno è metà del margine

esterno e il margine superiore è la metà del margine inferiore. La figura 20.4 permette di seguire la costruzione su un foglio con le proporzioni della carta A4; nella costruzione della figura (a) viene usato il rapporto $2 : 1$ tra la larghezza del testo e la larghezza complessiva dei margini, così come tra la larghezza del margine esterno rispetto al margine interno, e lo stesso viene fatto per le altezze. Quindi sei strisce per il testo e tre complessivamente per i margini, delle quali due per il margine maggiore e una per il margine minore

Sempre usando le strisce, ma con un rapporto diverso, si può ottenere una costruzione differente, ma che conserva la stessa proporzione fra le dimensioni della pagina e quelle della gabbia. Per esempio se si sceglie il rapporto $3 : 2$, occorrono complessivamente 25 strisce: 4 per il margine piccolo, 6 per il margine grande, 10 in totale per i margini, 15 per la gabbia; questo è quello che si è fatto nella figura 20.4(b).

Sviluppando i calcoli in modo algebrico, senza fare riferimento alle strisce (che diventerebbero infinite) si potrebbe calcolare la divisione della pagina con il rapporto aureo; il margine piccolo sta a quello grande come $1 : \phi$; e la somma dei margini sta al testo ancora come $1 : \phi$. Quindi il margine piccolo sta a quello grande che sta al testo come la proporzione $1 : \phi : \phi(1 + \phi)$. Svolgendo i calcoli, si ottiene che se w è la dimensione orizzontale o verticale della pagina rifinita, allora le dimensioni orizzontali o verticali rispettivamente dei margini e della gabbia sono $0{,}145\,898w$ per il margine piccolo, $0{,}236\,068w$ per il margine grande e $0{,}618\,034w$ per la gabbia. Ciò è quanto si vede nella figura 20.4(c).

La situazione è diversa per il disegno (h). In questo disegno si impone che la gabbia sia un rettangolo aureo, quindi la base della gabbia sta alla sua altezza come 1 sta a ϕ. Il foglio è un rettangolo UNI (A4) con rapporto fra la base e l'altezza pari a $\sqrt{2}$; se si impone il rapporto aureo anche fra i margini piccoli e quelli grandi, questo può essere fatto indipendentemente dalle dimensioni della gabbia. Quindi è possibile imporre, per esempio che la base della gabbia abbia il rapporto aureo rispetto alla somma dei margini laterali, ma non si può imporre la stessa relazione fra l'altezza della gabbia e la somma dei margini superiore e inferiore. Infatti, eseguiti i calcoli[4], si ottiene il disegno (h) che è piuttosto diverso dal disegno (c).

Nei primi tre schemi della figura 20.4 i rapporti numerici indicati sono sempre costituiti da due successivi numeri di Fibonacci[5]; è noto che la sequenza di Fibonacci ha il rapporto fra due successivi termini che tende al rapporto aureo; non è un caso che la figura 20.4(c) assomigli molto alla figura 20.4(b), benché in questa il rapporto $3 : 2 = 1{,}5$ non sia tanto vicino al rapporto aureo $1{,}618\,034\ldots$; ma la costruzione della figura 20.4 basata sulle strisce consente anche rapporti formati da numeri che non appartengono alla sequenza di Fibonacci. Se si sceglie

[4] I calcoli danno:

base del foglio	210,000 mm	margine interno	30,639 mm
altezza del foglio	297,000 mm	margine esterno	49,574 mm
base della gabbia	129,787 mm	margine superiore	33,231 mm
altezza della gabbia	210,000 mm	margine inferiore	53,769 mm

[5] La sequenza di Fibonacci comincia con 1, 1 e prosegue con ogni elemento costituito dalla somma dei due precedenti, quindi: 1, 1, 2, 3, 5, 8, 13, ...

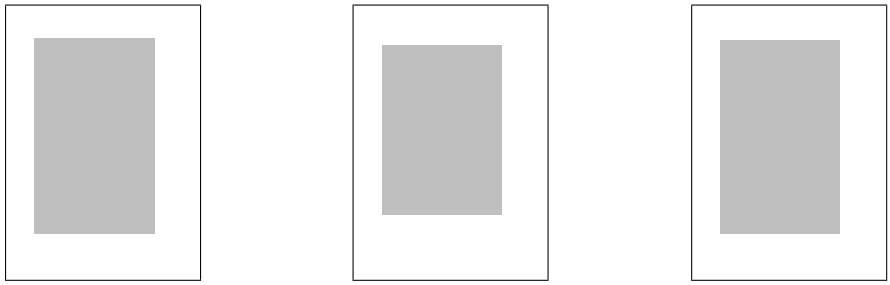


Figura 20.5: Confronto fra tre gabbie che sulla carta UNI A4 tengono conto del rapporto aureo. (*a*) gabbia ottenuta con il pacchetto `layaureo`; (*b*) gabbia con proporzioni auree fra le dimensioni della gabbia e i corrispondenti margini piccolo e grande; (*c*) gabbia aurea con margini aurei

il rapporto 4 : 3 le strisce occorrenti sarebbero 49 e si omette di disegnarle; si ottiene la figura 20.4(*d*) e, come si vede, le differenze non sono enormi.

Visto che in diverse figure si sono presentati dei disegni della pagina con margini aurei, si è riportata ancora la figura 20.5 dove sono messi a confronto i tre schemi, ripetendone le caratteristiche:

(*a*) In questo schema prodotto dal pacchetto `layaureo` la gabbia ha la base pari alla lunghezza ottimale in base alla regola di Bringhurst, aumentata di 4 pc – secondo la tabella 20.2 la giustezza conterrebbe 75 caratteri del font Latin Modern; l'altezza in rapporto aureo con la base; il margine interno e il margine esterno in rapporto aureo fra di loro; il margine superiore $\sqrt{2}$ volte più piccolo del margine inferiore. La pagina è A4 e la larghezza della gabbia è di 31 pc, per il font da 10 pt, come specificato sopra. Secondo il pacchetto `layaureo` la gabbia risulterebbe linearmente più grande di circa il 6% per il corpo di 11 pt e di un altro 6% più grande per il corpo di 12 pt.

(*b*) In questo schema i margini, sia in orizzontale sia in verticale, sono in proporzione aurea e la loro somma è in proporzione aurea con la dimensione orizzontale o verticale della gabbia; il rettangolo della gabbia ha quindi le stesse proporzioni della pagina.

(*c*) In questo schema, invece, il margine interno è in proporzione aurea con il margine esterno e la loro somma in proporzione aurea con la base della gabbia; l'altezza della gabbia è in proporzione aurea con la base; il margine superiore e il margine inferiore sono in proporzione aurea fra di loro, ma non con l'altezza della gabbia.

In questi tre esempi la giustezza è sempre circa 4 pc (circa 17 mm) più grande di quella raccomandata dalla regola di Bringhurst, ma probabilmente solo di cinque o sei caratteri in più, quindi una misura assolutamente accettabile. Si vede ad occhio che, almeno per la composizione in corpo di 10 pt, le gabbie (*a*) e

(c) sono praticamente identiche così come lo sono i margini. Con lo schema (a) usando un corpo maggiore per il font normale si avrebbe un riempimento della pagina significativamente maggiore.

Come si vede solo gli schemi (a) e (c) hanno la gabbia con la base e l'altezza in proporzione aurea, mentre nello schema (b) la gabbia ha la base e l'altezza in proporzione di $\sqrt{2}$. Nello schema (a) la base della gabbia è determinata in base alla regola di Bringhurst, mentre negli schemi (b) e (c) la base è determinata in base alla proporzione aurea rispetto alla somma dei margini. Le tre filosofie si rifanno tutte e tre al numero aureo, ma ne fanno un uso completamente diverso. Sta al compositore scegliere quella delle tre soluzioni che considera più gradevole, come sta a lui decidere se rifarsi al numero aureo o al metodo delle strisce.

Alcuni pacchetti determinano la gabbia e i margini riferendosi alle strisce, usando sempre una striscia per il margine piccolo, e due strisce per il margine grande, ma senza imporre nessun rapporto particolare fra le dimensioni della gabbia e quelle dei margini, lasciando, quindi, libera la scelta del numero delle strisce; nei disegni, (e), (f), (g), della figura 20.4 sono rappresentate, sempre con riferimento ad un foglio A4, le gabbie che si ottengono rispettivamente con 11, 10 e 8 strisce (il disegno con nove strisce è il disegno (a)). Si vede dunque che il metodo delle strisce realizzato dal pacchetto `typearea` o per le 10 strisce dalla classe `octavo` è molto versatile; volendo, con numeri alti di strisce, si può stabilire anche un rapporto diverso fra le strisce da dedicare al margine piccolo rispetto al margine grande, ed è quello che i parametri opzionali al pacchetto `typearea` consentono di fare.

Si osservi infine che in tutti i disegni della figura 20.4, tranne il disegno (h), la gabbia ha sempre le stesse proporzioni del foglio; inoltre per costruzione i margini grandi stanno ai margini piccoli come l'altezza del foglio sta alla base. Per cui la gabbia del testo rimane sempre adagiata con la sua diagonale sulla corrispondente diagonale del foglio. L'unico disegno che si discosta da questa proprietà è per costruzione il disegno (h), e quindi la diagonale della gabbia non giace sulla diagonale del foglio. Tuttavia la differenza non è enorme, ma alla fin fine il disegno (h) presenta una maggiore copertura del foglio con margini non troppo piccoli e ben proporzionati.

Il metodo delle strisce permette di suddividere anche le strisce destinate ai due margini in modo diverso fra i margini verticali e quelli orizzontali; è quindi un metodo molto versatile. Tuttavia si richiama ancora il lavoro di Wilson dove in molti casi le suddivisioni fra i margini non rispettano affatto la regola delle strisce e non ci si avvicina affatto al rapporto aureo. Appaiono curiosi gli esempi in cui il margine interno è vistosamente maggiore del margine esterno. Un caso particolare, che ha la sua giustificazione, è il libro di Knuth, *Concrete Mathematics*, per il quale l'autore ha sviluppato anche la famiglia di font denominata Concrete Computer Modern, con la sigla CC. A parte l'interesse specifico per il contenuto, il layout della pagina presenta un margine interno vistosamente maggiore di quello esterno, ma anche pieno di note marginali sistemate in questo margine, dove riporta le battute dei suoi studenti in corrispondenza degli argomenti trattati. Questa libertà stilistica e di contenuto è forse tipica dello humor anglosassone,

ma certamente la lettura di quelle battute alleggerisce molto la difficoltà della materia. Ecco quindi che con uno scopo preciso in mente si possono 'violare' le regole dettate dalla tradizione.

20.3 Lo scartamento e i contrografismi verticali

Sarebbe desiderabile che l'altezza della gabbia, in particolare quella che contiene il testo corrente abbia uno scartamento costante e contenga un numero intero di righe. La cosa non è semplicissima da calcolare, perché la prima riga del testo, non dovendo essere distanziata da nessuna riga precedente, ha una altezza convenzionale indicata da `\topskip`, quasi sempre pari a 10 pt, mentre le distanze fra le righe di base di tutte le righe successive sono pari a `\baselineskip`. Per calcolare l'altezza del testo, `\textheight`, in modo che contenga un numero intero di righe, bisogna eseguire dei calcoli fra numeri interi, con una aritmetica che non è familiare a chi è abituato ai calcoli a mano, ma che viene benissimo quando si eseguono i calcoli a macchina; si tratta di eseguire le divisioni conservando la parte intera del quoziente e gettando via il resto.

Scelto l'avanzamento di riga ottimale `\normalbaselineskip` per il font di corpo normale prescelto (anche questa una operazione delicatissima, che occupa diversi capitoli in ogni libro di tipografia) si può eseguire il calcolo in questa maniera: dal calcolo della gabbia si sottrae lo spazio destinato alle testatine, e ai piedini e ai loro contrografismi se si è scelto che tutti questi elementi facciano parte della gabbia; si ottiene quindi una prima approssimazione di `\textheight` che chiameremo x_1, si toglie `\topskip`, che chiameremo y e si divide per `\normalbaselineskip`, che chiameremo z; il risultato di questa divisione in generale non è un numero intero, ma la divisione che i programmi del sistema TEX eseguono è una divisione intera, quindi il risultato è la parte intera del quoziente, che chiameremo N; l'altezza della scatola del testo corrente non sarà quindi x_1, come assunto in partenza, ma sarà il nuovo valore x_2 calcolato come segue:

$$x_2 = Nz + y \qquad \text{dove} \qquad N \text{ è la parte intera di } (x_1 - y)/z$$

Esercizio 20.2 Per un romanzo privo di testatina e con un piedino contenente solo il numero della pagina, e che quindi non si considera facente parte della gabbia, `\textheight` coincide con l'altezza della gabbia. Se si usa il metodo delle strisce e il rapporto 2 : 1 con una pagina rifinita di 170 mm per 240 mm, e si è scelto il font Computer Modern di corpo 10 pt e un avanzamento di riga normale di 12 pt, quanto deve valere `\textheight` perché contenga un numero intero di righe?

Si può procedere anche a rovescio: scelto lo scartamento che si vorrebbe usare, z_1, si calcola quante righe N_t *totali* stanno nell'altezza desiderata; poi si ricalcola il nuovo scartamento z_2 in modo che nell'altezza specificata stiano esattamente

le N righe così definite. Perciò i calcoli da fare sono i seguenti[6]:

$$M = N_{\mathrm{t}} - 1 = \left\lfloor \frac{x-y}{z_1} \right\rfloor \qquad \text{e poi} \qquad z_2 = \left(\frac{x-y}{M} \right)$$

L'ultima divisione deve essere una normale divisione fratta; è più facile da calcolare con una ordinaria calcolatrice, che non farla determinare a `pdftex`; in realtà *oggi* questo sarebbe perfettamente in grado di eseguire la divisione fratta senza problemi; oggi, infatti, le versioni moderne dei programmi di composizione del sistema TeX consentono di determinare nativamente il valore di alcune espressioni aritmetiche grazie alle estensioni di `etex` incorporate già da qualche anno. Si tralascia qui di mostrare come fare per far fare i calcoli al programma stesso, ma si indirizza il lettore alla documentazione di `etex` contenuta in `.../doc/etex/base/etex-man.pdf`, al paragrafo 3.5 intitolato significativamente *Expressions*; ci si ricordi però, che queste espressioni consentono di trasformare un numero fratto in un numero intero per arrotondamento, non per troncamento, quindi è importante ricordarsene per evitare di stupirsi per risultati diversi da quelli eseguibili a mano.

Dal 2020 il nucleo di LaTeX contiene anche le funzionalità del pacchetto *xfp* (che quindi non ha bisogno di essere caricato) che includono anche le operazioni di aritmetica intera; di nuovo bisogna ricordare che per la divisione intera viene fatto l'arrotondamento all'intero più vicino e, per esempio, 2,5 viene arrotondato a 3.

Nel presente testo lo scartamento è di 12 pt per il corpo normale di 10 pt; il file `bk10.clo`, corrispondente all'opzione *10pt*, determina l'altezza del testo con una prima approssimazione pari all'altezza del foglio diminuita di 3,5 pollici, un valore forfettario che tiene conto dei margini, della testatina e del suo spazio di separazione dal testo, della prima riga di testo, ma non del piedino; poi divide il risultato per l'avanzamento di riga normale conservando la parte intera del risultato; infine rimoltiplica l'avanzamento normale per questo numero intero a alla fine aggiunge `\topskip`. Praticamente quel che si è indicato con il primo procedimento.

Sarebbe desiderabile che le righe di testo corrente che si vedono controluce fra il recto e il verso della pagina siano appoggiate sulle stesse linee di base; in questo modo si riduce l'effetto della semitrasparenza della carta, che provocherebbe un disturbo alla lettura se ciò non avvenisse.

Bisognerebbe quindi che ogni struttura del testo stampato che compare su ogni pagina, dalla semplice riga di testo, ai titolini dei paragrafi e delle loro suddivisioni, dalle formule in display alle inserzioni di figure, tabelle o altro, fossero sempre anche loro dei multipli interi dell'avanzamento di riga normale. Questo è molto difficile da ottenere, anche se si prendono tutte le misure e le precauzioni possibili e immaginabili per avere strutture fatte come si è detto. Nei testi letterari ci si riesce abbastanza bene, ma in testi tecnici è difficilissimo. Anche i compositori che lavorano con programmi di impaginazione professionali

[6] In matematica il troncamento all'intero n di un numero reale x si indica con $n = \lfloor x \rfloor$.

trovano serie difficoltà e nella migliore delle ipotesi devono aggiustare a mano l'altezza di ogni struttura che non sia costituita da semplice testo composto con il font di default nel corpo normale.

In pratica bisogna determinare gli spazi bianchi, i contrografismi, in modo che la loro altezza sommata a quella delle strutture in evidenza sia sempre un multiplo intero dell'avanzamento di riga normale. Questo non è difficilissimo da fare ma richiede una pazienza infinita. Sembra, però che il mark up ConTEXt che usa come interprete il programma luatex riesca a comporre in questo modo senza che l'utente debba intervenire a mano. Chi scrive non ha esperienza in merito, ma il problema costituisce l'argomento di diversi articoli nella rivista ArsTEXnica del GUIT.

Il punto è che bisogna fare i conti con l'algoritmo che i programmi di composizione del sistema TEX usano per suddividere il testo in pagine. Come si può bene capire, l'interprete non lascia il titolo di un paragrafo in fondo ad una facciata per cominciare il testo del paragrafo nella facciata successiva (molti word processor di tipo WYSIWYG lo fanno!); l'interprete cerca anche di evitare le righe orfane e le righe vedove. Una riga orfana è una riga isolata in fondo alla pagina quando il capoverso prosegue nella pagina successiva ("le righe orfane non hanno un passato ma hanno un futuro" è la frase per ricordare di che cosa si tratta); una riga vedova è una riga isolata all'inizio di una pagina il cui capoverso è cominciato nella pagina precedente ("le righe vedove hanno un passato ma non hanno un futuro").

Questo significa che un paragrafo con il suo titolo corrente può cominciare a 4 o 5 righe equivalenti di testo corrente dal fondo di una pagina, perché deve contenere il suo titolo corrente, il suo contrografismo e almeno due righe di testo, altrimenti la sua sola prima riga resterebbe orfana. Analogamente la fine di un capoverso non può avvenire mandando a capo in una nuova pagina la sua ultima riga, perché questa resterebbe vedova. Quando c'è della matematica in display si possono avere altre possibilità di righe orfane o vedove prima o dopo la formula. Quando dunque si devono evitare le righe orfane o vedove si hanno due possibilità: o si compone senza giustificare il piede delle pagine affacciate, oppure si giustificano questi piedi, ma si introduce dello spazio in più nei contrografismi o fra un capoverso e l'altro; in sostanza si disturba la griglia delle righe.

Una soluzione sarebbe quella di accorciare di una riga due pagine affacciate (uno *spread*), in modo che la riga vedova che apparirebbe nella pagina di destra senza l'accorciamento suddetto, verrebbe preceduta dall'ultima riga dalla pagina di sinistra riportata a destra grazie all'accorciamento; nello stesso tempo, scorciando anche la pagina di destra le due griglie dello *spread* restano allineate. Se invece la riga vedova gira nella pagina di sinistra, la prima riga di uno *spread*, accorciando di una riga le due pagine dello spread precedente, la vedova non resta più tale perché verrebbe preceduta da due righe provenienti dallo *spread* precedente. Queste correzioni, come è facile immaginare, vanno fatte all'ultimo, perché qualunque modifica del testo potrebbe distruggere tutta la paziente operazione appena descritta.

Certo, in fase di rifinitura finale, si può intervenire inserendo in punti strategici opportuni comandi di `\newpage`, che eliminano la giustificazione al piede delle pagine, oppure comandi `\enlargethispage` con o senza asterisco che ugualmente possono eliminare la giustificazione al piede delle pagine affacciate. Questo fatto non ha nulla a che vedere con il disegno grafico, ma bisogna sempre ricordarsene per capire perché un disegno accuratamente pensato possa deludere le aspettative.

Un'altra tecnica poco conosciuta per gestire la lunghezza dei capoversi per allungarli o per accorciarli, senza modificarne il testo, consiste nello specificare il comando primitivo `\looseness` alla fine del capoverso insieme ad un valore intero positivo o negativo: per esempio `\looseness1` serve per consentire l'allungamento del capoverso di una riga. Questa tecnica è efficace: (*a*) se il capoverso è sufficientemente lungo; (*b*) se il capoverso è da allungare e il suo ultimo rigo è sufficientemente lungo; (*c*) se il capoverso è da accorciare e il suo ultimo rigo è sufficientemente corto. Se queste condizioni non sono verificate il comando `\looseness` lascia le cose come stanno e quindi non risulta efficace, ma non disturba la composizione. In questa guida l'artificio indicato è stato usato una mezza dozzina di volte.

20.4 I capoversi

Una delle caratteristiche che più distinguono le varie composizioni tipografiche riguarda lo stile di composizione dei capoversi.

In generale sarebbe preferibile che ogni capoverso non fosse separato dal capoverso precedente da una interlinea. Con LaTeX generalmente questa separazione dei capoversi con una interlinea è attuata solamente all'interno delle liste semplici, o delle enumerazioni o delle descrizioni. In questi casi, visto che il testo ha già un margine sinistro rientrato, il tipico rientro del capoverso sarebbe probabilmente eccessivo.

Invece i capoversi del testo 'normale' non sono separati da una interlinea, ma hanno la prima riga rientrata. Questo rientro serve per marcare l'inizio di un capoverso ed è particolarmente utile quando il capoverso precedente ha il suo ultimo rigo che termina praticamente a filo del margine destro. Si capisce subito, allora, che il primo capoverso di un paragrafo non ha bisogno di nessun rientro, perché non è direttamente preceduto da nessun altro capoverso, ma solo dal titolo del paragrafo.

Nella tradizione compositiva francese, invece, anche il primo capoverso di un paragrafo è rientrato. Nell'Ottocento e fino alla prima metà del Novecento anche la tipografia italiana seguiva la tradizione francese, non solo per quel che riguarda il rientro del primo capoverso, ma anche per le spaziature davanti ai segni di interpunzione alti e anche fra i caporali e il testo fra loro contenuto. Se si desidera seguire la tradizione francese di rientrare anche il primo capoverso di un paragrafo basta caricare il pacchetto *indentfirst*.

Il rientro del capoverso dovrebbe essere costante, indipendentemente dal corpo usato, per esempio 10 pt. Se si specifica il rientro mediante una unità

di misura dipendente dal font in uso, per esempio 1 em, questo coincide con 10 pt con il corpo 10, ma aumenta o diminuisce con il corpo del font con cui il capoverso viene composto; questo potrebbe essere un inconveniente estetico perché se si usano spesso incisi composti in corpo minore, i rientri dei capoversi non appaiono più incolonnati. Si consiglia quindi, nell'usare i pacchetti che consentono di modificare il layout della pagina, di definire il rientro mediante una misura assoluta, per esempio lo stesso ammontare dell'avanzamento di riga per il testo normale, da 12 pt a 15 pt.

* * *

NON SEMPRE IL PRIMO CAPOVERSO è preceduto dal titolo del paragrafo. Dipende dal contesto, ma in diversi tipi di documenti a stampa o in certe loro sezioni, come per esempio una introduzione, non si divide il testo in paragrafi, benché vengano trattati diversi argomenti fra di loro non tanto correlati, tanto da dover essere sezionati con i rispettivi titoli. Spesso si ricorre a delle decorazioni.

Nei libri moderni le decorazioni sono spesso assenti; altrettanto spesso si riducono a una breve sequenza di simboli, per esempio tre asterischi centrati, come alla fine del paragrafo precedente (terminato con i tre asterischi) e prima del capoverso precedente (il primo del paragrafo corrente, iniziato con un capolettera), e vengono usati per separare verticalmente parti di testo che non richiedono un sezionamento vero e proprio, ma nello stesso tempo non sono propriamente contigue da un punto di vista concettuale.

Esistono font che contengono sequenze di motivi floreali o motivi geometrici con i quali si possono realizzare decorazioni più eleganti, ma che spesso stonano nei libri moderni.

Se invece si deve riprodurre un libro antico, o anche solo vecchiotto, si può ricorrere a questi font e si possono cercare su CTAN i pacchetti per gestire queste decorazioni nel modo più efficace; tra gli altri si segnala il pacchetto *fourier*.

Al contrario anche in alcuni libri moderni si usano i capilettera; queste sono lettere di corpo decisamente maggiore di quello del testo circostante e vengono usate rialzate o ribassate per marcare l'inizio di un capoverso, o, meglio ancora, il capoverso iniziale di un capitolo e/o di un paragrafo; anticamente si usavano anche capilettera decisamente ornati e di corpo diverse volte maggiore del corpo normale, per esempio in corpo 70 pt quando si compone in corpo 10 pt.

Per usare i capilettera (*versal* in inglese e *lettrine* in francese) si può agevolmente usare il pacchetto *lettrine* con il quale sono disponibili comandi e opzioni di diverso genere per collocare i capilettera nella posizione desiderata scegliendone il font, le dimensioni e gli altri parametri stilistici. Vale la pena di leggerne la documentazione in `.../doc/latex/lettrine/lettrine.pdf`, ma è molto interessante esaminare gli esempi riportati in `.../doc/latex/lettrine/demo.pdf`.

È importante considerare anche l'effetto che si ottiene quando si comincia un capoverso con un capolettera e si prosegue con il maiuscoletto per qualche parola, per poi tornare al font normale; questo è un espediente che viene usato spesso quando si usano i capilettera, per mitigare la transizione fra le grandi

dimensioni della lettera iniziale e il corpo normale; è esattamente quello che si è fatto all'inizio di questo paragrafo.

20.5 Testatine e piedini

Delle testatine e dei piedini si è già commentato abbastanza sul fatto che intervengano o non intervengano nella definizione della gabbia; si è anche detto del pacchetto *fancyhdr* per mettere nella forma dovuta testatine e piedini.

Non si ritorna su questi argomenti, ma ci si sofferma sulla sostanziale differenza che per il sistema TeX esiste fra una testatina e un piedino; non si tratta semplicemente di due blocchetti di testo, generalmente composti di una sola riga ciascuno, messi uno in testa e uno al piede del testo adeguatamente spaziati da questo e, talvolta, adeguatamente separati da semplici decorazioni, in generale formate da un sottile filetto orizzontale.

Il punto è che lo spazio di separazione fra la testatina (eventualmente comprensiva del suo filetto) e il testo sottostante è costituito da spazio vero e proprio di una dimensione ben specificata dal grafico editoriale. Invece lo spazio specificato per distanziare il piedino dal piede del testo rappresenta lo scartamento fra la linea di base dell'ultima linea del testo (o dal fondo del rettangolo ideale che lo contiene) e la linea di base della prima o unica riga del piedino.

In buona sostanza, se si specifica che la distanza `\headsep` fra l'`\header` (testatina) e il testo è di 10 mm, una volta composta una pagina con quella testatina si prende il doppio decimetro, si misura e si rilevano esattamente 10 mm.

Se si specifica che la spaziatura del piedino `\footskip` sia di 10 mm, e, attenzione, non c'è nulla da specificare per l'altezza del piedino, dopo aver stampato una pagina piena con quel piedino si misurano i 10 mm fra le linee di base dell'ultima riga del testo e della prima o unica riga del piedino, si trovano quei 10 mm; ma, considerando i discendenti dell'ultima riga e il fatto che la prima o unica riga del piedino giace sopra la sua linea di base, la distanza effettiva fra il piedino e il testo risulta decisamente minore di 10 mm e questa differenza è otticamente ben percepibile. D'altra parte si è già mostrato lo schema quotato delle due pagine affacciate sinistra e destra nella pagina 170 e nella successiva: si vede chiarissimamente che la freccia che indica la misura di `\footskip` va dalla base del testo alla base del piedino, mentre la freccia che indica la misura di `\headsep` va dalla base della testatina al colmo del testo.

Di questo fatto bisogna tenere ben conto nel progettare la geometria della pagina; certamente i pacchetti impiegati per determinare i parametri geometrici aiutano, ma non si può fare a meno di eseguire alcune prove di stampa per valutare gli effetti; questi ultimi saranno tanto più vistosi quanto più 'ingombrante' è il piedino; un semplice numero di pagina centrato nel piedino ma un po' troppo vicino al testo sovrastante disturba la vista, ma lo si può sopportare; se invece lo stesso piedino contiene a sinistra il nome dell'autore, al centro il numero della pagina e a destra il nome della collana di pubblicazioni a cui appartiene

il rapporto corrente, allora un simile piedino troppo vicino al testo disturba davvero e non lo si può accettare.

Per avere lo schema metrico delle due pagine affacciate lo si riporta di nuovo nella pagina 536 e nella successiva in modo da avere a disposizione più facilmente le varie misure della pagina descritte in questo capitolo. Ci si ricordi che le quote indicate sono relative alla classe *book* con un formato della pagina B5, ma senza avere modificato le dimensioni predefinite. Su fogli della serie ISO la disposizione del testo non è perfetta, ma è accettabile, specialmente se si fa un largo uso di note a margine. Si capisce quindi perché la maggior parte degli utenti ricorre ai pacchetti di personalizzazione, specialmente *geometry* e *typearea*, per disporre di una maggiore flessibilità nel disegno della pagina[7]. Questo è bene, ma non bisogna esagerare: quei pacchetti eseguono al posto nostro i vari calcoli necessari e seguono le nostre indicazioni stilistiche come meglio possono, ma sta al compositore valutare il risultato da un punto di vista estetico.

20.6 I pacchetti di personalizzazione

20.6.1 Il pacchetto *geometry*

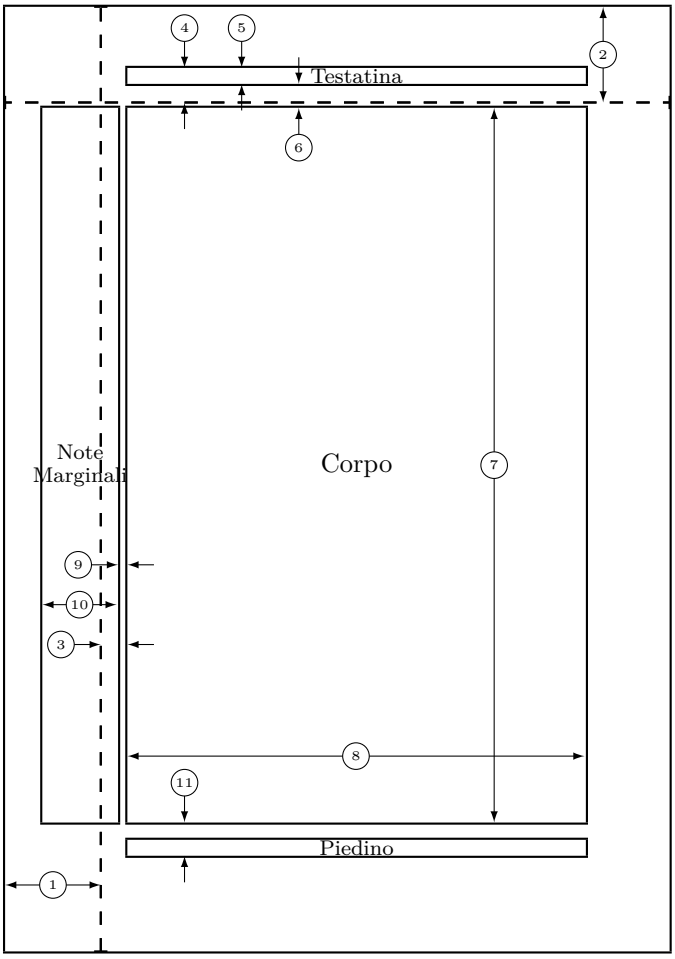
Il pacchetto *geometry* è un pacchetto che contiene tutte le necessarie opzioni e macro per descrivere il layout geometrico di una pagina 'normale', tenendo anche conto dell'ingombro ottico delle testatine e dei piedini; permette di specificare i margini, di scegliere un layout classico, di scegliere le proporzioni adeguate allo stile desiderato e al font usato. Esegue, insomma, tutte le operazioni descritte nei paragrafi precedenti.

Date le molteplici applicazioni, la sua documentazione è piuttosto importante; la si trova in `.../doc/latex/geometry/manual.pdf`.

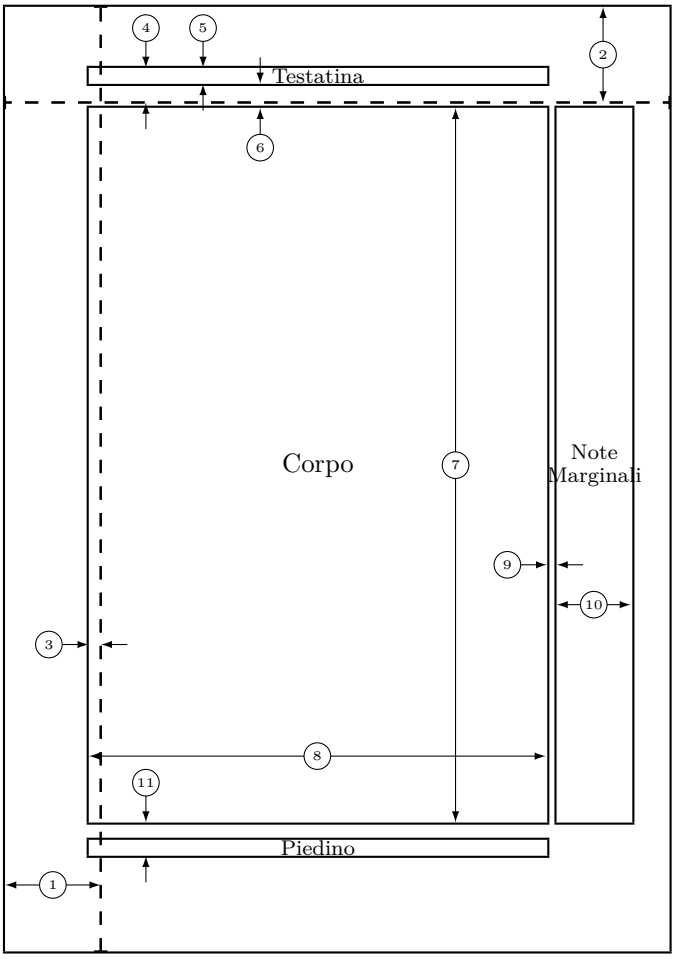
Una delle cose più utili del pacchetto è la sua capacità di determinare le informazioni geometriche mancanti a partire da un certo numero di parametri di default. Per esempio esso usa un rapporto predefinito (che ovviamente può essere modificato dal compositore) fra il margine interno (o quello superiore) e il margine esterno (o quello inferiore) delle pagine; se il documento è stampato solo in bianca, il rapporto di default viene usato solo per i margini superiore e inferiore, mentre per quello interno ed esterno viene usato il rapporto $1:1$, vale a dire che la gabbia viene centrata orizzontalmente fra margini uguali. Se invece il documento viene stampato in bianca e volta il rapporto predefinito o specificato dal compositore viene usato per tutti i margini. Conoscendo le dimensioni della pagina rifinita, il pacchetto è in grado di calcolare la gabbia e i margini a partire da una sola di queste informazioni.

Per esempio, il manuale presenta questo caso: supponiamo di avere una pagina finita in formato A4, quindi di 210 mm per 297 mm, e di specificare il

[7] La classe *memoir* contiene i suoi propri comandi per la creazione della geometria della pagina; essi sono molto flessibili e potenti; *memoir* mette anche a disposizione dei comandi per presentare graficamente la disposizione dei blocchi di testo, i titoli, e altre cose del genere in modo da aiutare il compositore a scegliere meglio i parametri geometrici delle varie pagine.



1	un pollice + \hoffset	2	un pollice + \voffset
3	\evensidemargin = 20pt	4	\topmargin = -26pt
5	\headheight = 12pt	6	\headsep = 18pt
7	\textheight = 538pt	8	\textwidth = 345pt
9	\marginparsep = 7pt	10	\marginparwidth = 57pt
11	\footskip = 25pt		\marginparpush = 5pt (non mostrato)
	\hoffset = 0pt		\voffset = 0pt
	\paperwidth = 500pt		\paperheight = 711pt



1	un pollice + \hoffset	2	un pollice + \voffset
3	\oddsidemargin = -9pt	4	\topmargin = -26pt
5	\headheight = 12pt	6	\headsep = 18pt
7	\textheight = 538pt	8	\textwidth = 345pt
9	\marginparsep = 7pt	10	\marginparwidth = 57pt
11	\footskip = 25pt		\marginparpush = 5pt (non mostrato)
	\hoffset = 0pt		\voffset = 0pt
	\paperwidth = 500pt		\paperheight = 711pt

margine esterno di 24 mm; il rapporto predefinito fra i margini vale 2 : 3 (come nella figura 20.4(b)) e il riempimento predefinito della pagina vale 0,7 (un po' maggiore rispetto al valore di 3/5 che appare nella figura 20.4(b)). Allora:

$$\begin{aligned}
\text{margine interno} &= (2/3) \times 24\,\text{mm} = 16\,\text{mm} \\
\text{giustezza orizzontale} &= 210\,\text{mm} - 24\,\text{mm} - 16\,\text{mm} = 170\,\text{mm} \\
\text{giustezza verticale} &= 0{,}7 \times 297\,\text{mm} = 207{,}9\,\text{mm} \\
\text{margine superiore} &= \frac{1}{1+3/2} \times (297\,\text{mm} - 207{,}9\,\text{mm}) = 35{,}64\,\text{mm} \\
\text{margine inferiore} &= (3/2) \times 35{,}64\,\text{mm} = 53{,}46\,\text{mm}
\end{aligned}$$

Allo stesso modo si sarebbe potuta specificare qualunque altra singola dimensione (oltre alle dimensioni della pagina finita) e si sarebbero potute calcolare le misure mancanti. *geometry* si incarica poi di tradurre i valori di giustezze e di margini in termini di lunghezze interne, quelle usate dall'interprete per eseguire la composizione tipografica.

Non si insiste oltre, perché la funzionalità del pacchetto è talmente vasta che continuando con gli esempi si finirebbe con riscrivere il manuale. Ciò non impedisce di ribadire che il pacchetto *geometry* è uno dei pacchetti preferiti da ogni compositore che voglia personalizzare un pochino i suoi documenti senza conformarsi agli stili predefiniti dalle classi standard della distribuzione del sistema TeX. Come è già stato notato, le funzionalità di *geometry* sono tutte incorporate nella classe *memoir* che le estende ulteriormente; se si usa questa classe, non è necessario caricare il pacchetto *geometry* a meno che non se ne vogliano sfruttare i comandi, invece di affidarsi ai comandi di *memoir* che hanno nomi diversi. Chi scrive ha usato sia l'una che l'altra strada e la sua opinione è che se si usa *memoir* è meglio non caricare né *geometry* né *typearea* (vedi sotto), e usare solo i comandi e le altre funzionalità della classe.

Anche la collezione KOMA-script dispone di un insieme di comandi per specificare la geometria della pagina; anzi, questi comandi sono contenuti nel pacchetto *typearea* che può essere caricato separatamente anche quando si usino classi non appartenenti alla collezione KOMA-script. Vale la pena di documentarsi nella guida `scrguien.pdf` (in inglese), in `.../doc/latex/koma-script`, per vedere che cosa offra il pacchetto *typearea* il quale, di default, usa il metodo delle strisce per la specificazione della geometria della pagina; naturalmente sono disponibili comandi accessori per eseguire delle variazioni sul layout basato sulle strisce; in ogni caso il procedimento di calcolo è molto flessibile.

20.6.2 I titoli e i titolini: i pacchetti *titlesec*, *fancyhdr*, *sectsty* e *tocloft*

Per definire lo stile compositivo dei titoli correnti (titolini) ci sono diversi pacchetti, sempre senza trascurare la classe *memoir* che include in sé la maggior parte della funzionalità dei vari pacchetti presenti nell'installazione completa del sistema TeX.

Uno di quelli maggiormente citati è *titlesec*, il cui manuale a sua volta si premura di avvisare i possibili utenti che se desiderano usare strumenti più semplici possono usare i pacchetti *fancyhdr*, *sectsty* e *tocloft*. Sono tutti pacchetti validissimi, in effetti, ma ognuno è destinato a modificare lo stile solo delle testatine, oppure dei titoli correnti, oppure degli indici[8]. Invece *titlesec* dovrebbe poter agire su tutti i fronti con una interfaccia unificata.

Per quel che riguarda l'aspetto da dare ai titolini, il pacchetto può semplicemente essere invocato con alcune opzioni; queste controllano la famiglia, la serie e la forma dei font; si può stabilire il corpo usato nei titolini come anche la loro composizione giustificata, centrata, o in bandiera. Si può facilmente scegliere come comporre l'etichetta identificativa della sezione sia per quel che contiene sia per il font con cui è scritta.

Se si vogliono fare personalizzazioni più estese il pacchetto offre una moltitudine di comandi per specificare ogni possibile dettaglio relativo a titoli e titolini. Naturalmente queste cose sono applicabili non solo a titoli correnti di ogni sezione ma anche alle testatine e ai piedini.

Caricando il pacchetto *xcolor* si possono usare anche i colori per tutte queste 'iscrizioni' particolari. Naturalmente sta al buon gusto del compositore o del grafico editoriale decidere se usare i colori e sceglierli in modo adeguato anche al tipo di documento che viene composto. Negli anni '90 era abbastanza frequente l'uso di un colore celeste sia per i titolini, sia per le formule, sia per le figure; era un tentativo per evitare le fotocopie, visto che le fotocopiatrici di quell'epoca avevano difficoltà a copiare alcuni colori, fra i quali il celeste. Oggi il celeste può ancora essere usato (magari non così chiaro come allora) ma certamente non per evitare le fotocopie perché oggi le fotocopiatrici copiano perfettamente qualsiasi colore.

I comandi estesi per affrontare personalizzazioni importanti nello stile della pagina servono anche per definire nuovi stili o per modificare quelli già definiti. In particolare è possibile definire variazioni di stile rispetto alla pagine normali per le pagine che contengono solo oggetti flottanti, nelle quali si possono usare altre iscrizioni nelle testatine e/o nei piedini, e si possono eventualmente eliminare i filetti normalmente presenti affinché non possano essere confusi con oggetti appartenenti agli oggetti flottanti.

Si possono gestire anche i 'mark'; i programmi di composizione del sistema TEX lavorano con tre speciali marcatori per definire che cosa deve essere scritto nelle testatine e/o nei piedini. Sono il `\topmark`, `\botmark` e `\firstmark`; essi sono caricati con il ⟨*titolo breve*⟩ dei comandi `\chapter`, `\section` (ed eventualmente `\subsection` con la classe *article*) e consentono di avere sulla pagina corrente o sulle pagine di destra l'informazione del titolo del paragrafo (o del sottoparagrafo) corrente per la pagina nella quale compare l'indicazione; se nel corso della pagina corrente viene iniziato un nuovo paragrafo, a seconda del tipo di documento può essere importante mostrare nella testatina, per esempio, il titolo del paragrafo

[8] È opportuno aggiungere anche il pacchetto *caption* che consente di personalizzare le didascalie degli oggetti flottanti.

che era in vigore all'inizio della pagina oppure quello in vigore alla fine della pagina; tutte queste informazioni vengono gestite diversamente dai programmi di composizione, ma in entrambi i casi esse sono limitate a quei tre marchi; LaTeX li gestisce attraverso entità che si chiamano `\leftmark` e `\rightmark` che vengono 'caricati' mediante l'uso implicito di `\markboth` e `\markright` attraverso i comandi di sezionamento; il processo è abbastanza schermato all'utente normale, che così non deve preoccuparsi di queste cose piuttosto delicate. Con il pacchetto *titlesec* si hanno molte più possibilità perché si dispone di più mark e di più flessibilità nei campi delle testatine e dei piedini. I programmi recenti, che contengono le estensioni di etex, permettono di disporre di molti più mark, con i quali si possono ottenere effetti speciali piuttosto interessanti, che però sono alla portata solo dei programmatori che scrivono le classi o in file di estensione per il linguaggio LaTeX.

Il pacchetto consente anche di comporre indici parziali; per esempio, all'inizio di ogni capitolo si potrebbero stampare l'indice interno del capitolo e anche gli elenchi delle tabelle e/o delle figure contenute nel capitolo; naturalmente questi indici parziali possono essere composti con uno stile diverso dall'indice generale; tuttavia, anche in questo caso, l'uso di questi indici parziali deve essere adeguatamente pensato in funzione del particolare documento che viene composto. Non vale il principio 'italico' che "siccome lo si può fare, allora lo si deve fare"!

Vale quanto viene detto nel paragrafo 8 della documentazione:

> Avendo letto questa documentazione dovrebbe essere chiaro che questo non è un pacchetto per l'utente occasionale a cui piace il layout standard e vuole fare qualche piccola modifica. Questo è invece uno strumento per il tipografo professionale che ha una idea precisa del layout che vuole realizzare ma non ha le conoscenze [informatiche] per farlo. Non si è fatto nessuno sforzo per migliorare il vostro gusto nella composizione dei titolini.[9]

20.6.3 Testatine

La descrizione dello stile delle pagine fatta nel paragrafo precedente è piuttosto sommaria ed è meglio approfondire per poter rendersi conto esattamente come sono composte le testatine negli stili delle pagine `headings` o `myheadings` o in qualunque altro stile che il compositore si voglia definire.

I comandi per definire il contenuto delle testatine di sinistra e di destra sono contenuti nelle macro `\@evenhead` e `\@oddhead`, comandi interni, protetti con il segno `@`. Se il documento è composto solo in bianca, viene usata solo la testatina composta con lo stile delle pagine dispari; se il documento è composto in bianca e

[9] Once you have read the documentation it should be clear that this is not a package for the casual user who likes the standard layout and wants to make simple changes. This is a tool for the serious typographer that has a clear idea of what layout wants and doesn't have the skill to get it. No attempt is made to improve your taste in section formatting.

volta, vengono usate entrambe le macro, in particolare `\@oddhead` per le testatine delle pagine dispari, e `\@evenhead` per le testatine delle pagine pari.

Per la classe *book* queste due macro contengono (semplificato un po') il codice seguente:

```
\def\@oddhead{\textsl{\MakeUppercase{\rightmark}}\hfill\thepage}
\def\@evenhead{\thepage\hfill\textsl{\MakeUppercase{\leftmark}}}
```

Come si vede, e come si constata guardando la pagina corrente, le testatine hanno i titoli correnti composti con un font tondo inclinato e completamente in lettere maiuscole, grazie ai comandi `\MakeUppercase`; il numero della pagina viene composto all'estremità esterna delle testatine in carattere normale. Ma che cosa viene scritto nelle testatine? Per la classe *book*, che compone di default in bianca e volta, nella testatina di sinistra, pagina pari, viene inserito il titolo del capitolo, mentre nella testatina di destra, pagina dispari, viene inserito il titolo del paragrafo. Queste due informazioni sono contenute nelle macro `\rightmark` e `\leftmark`. A loro volta questi marchi sono inseriti quando si usano esplicitamente o implicitamente il comandi `\markboth` o `\markright`. La grande macro che definisce lo stile dalla pagina, per esempio `headings`, non definisce solo il contenuto delle testatine stesse, ma anche che cosa i comandi `\chapter` e `\section` caricano attraverso i comandi `\markboth` e `\markright` rispettivamente, tenendo conto del fatto che capitoli e paragrafi siano numerati e compaiano nel corpo del testo (`\mainmatter`) invece che nelle parti preliminari (`\frontmatter`) o nelle parti finali (`\backmatter`).

Quando si usa un comando `\chapter` nel corpo del testo, il suo titolo (breve) viene implicitamente usato nel comando `\markboth`, a sua volta chiamato dal comando `\chaptermark` che a sua volta viene chiamato da `\chapter`. Quando si usa lo stile di pagina `myheadings` i comandi `\markboth` e `\markright` vengono usati esplicitamente dal compositore. È inoltre possibile definire altri stili di pagina attraverso i pacchetti citati prima, oppure attraverso i comandi già presenti dentro alcuni file di classe, come *memoir*; infine possono venire usati i comandi primitivi simili a quelli che vengono usati nel nucleo di LaTeX per definire gli stili `headings` e `myheadings` a cui ci si può riferire per prendere spunto.

Bisogna capire bene come agiscono questi comandi di marcatura. È bene ricordare che LaTeX consiste di moltissime macro che durante la loro esecuzione vengono sviluppate fino ai comandi primitivi. Nella fattispecie i comandi primitivi relativi ai marchi sono `\firstmark`, `\topmark` e `\botmark`. Quando la routine di uscita di LaTeX compone la pagina completa di testatine, piedini, inserti flottanti, eccetera, fa riferimento ai marchi contenuti in `\firstmark` e `\botmark`; come suggerisce il loro nome, il primo rappresenta il marchio che si incontra all'inizio della pagina e il secondo è quello che si incontra alla fine della pagina. Invece `\topmark` è quello che ogni pagina eredita dall'ultimo marchio della pagina precedente, ed è quello che viene usato se la pagina non contiene un altro primo marchio. Se la pagina non contiene nessun nuovo marchio, tutti e tre sono uguali a `\topmark`, il cui valore viene passato per eredità alla pagina successiva e così

via. Dopo il primo marchio che LaTeX incontra nel comporre un documento, in tutte le pagine successive, che siano presenti o assenti nuovi marchi, i tre parametri sono sempre definiti.

Per rendere le cose un poco più flessibili, LaTeX usa i due comandi `\markboth` e `\markright` che inseriscono rispettivamente o due o una sola informazione in ogni marchio che viene messo nel codice da cui la routine di uscita preleverà le informazioni per le testatine. Precisamente il contenuto di ogni marchio è fatto di due informazioni raccolte fra parentesi graffe secondo lo schema seguente:

`{`⟨*marchio di sinistra*⟩`}{`⟨*marchio di destra*⟩`}`

dove le qualificazioni "di sinistra" o "di destra" si riferiscono al contenuto del gruppo di sinistra piuttosto che a quello di destra, ma non si riferiscono, è importante sottolinearlo, né alla pagina di sinistra, né alla pagina di destra. Di solito, dunque, il titolo di un capitolo forma il ⟨*marchio di sinistra*⟩ e il titolo del paragrafo forma il ⟨*marchio di destra*⟩.

Ecco la spiegazione del meccanismo: il comando `\rightmark` preleva dal *primo* marchio nella pagina, `\firstmark` oppure `\topmark`, la componente di destra di quel marchio; invece `\leftmark` preleva la componente di sinistra dell'ultimo marchio presente nella pagina, `\botmark`.

Questo spiega perché, se in una pagina non compare l'inizio di nessun paragrafo o un dato paragrafo inizia a metà pagina, il titolo corrente nella testatina è quello corrispondente al paragrafo che era ancora in vigore quando la pagina è iniziata (marchio ereditato dalle pagine precedenti), non all'eventuale paragrafo iniziato a metà pagina. Se si desiderasse l'indicazione del paragrafo che è in vigore alla fine della pagina, non bisognerebbe prenderlo dal `\firstmark` o dal `\topmark`, ma dal `\botmark`.

Ecco perché se si volesse indicare nella testatina di un dizionario il primo e l'ultimo lemma che compaiono nella pagina bisognerebbe innanzitutto definire il comando per inserire il lemma con, per esempio:

```
\newcommand*{\lemma}[1]{#1\markboth{#1}{#1}
```

e poi inserire il ⟨*lemma*⟩ con:

`\lemma{`⟨ *lemma*⟩`}`

Bisogna definire un nuovo stile di pagina in modo che le testatine di sinistra sia quelle di destra non contengano il numero della pagina, il quale va quindi messo nel piedino, ma contengano i riferimenti al primo lemma in vigore all'inizio della pagina e l'ultimo lemma inserito nella pagina; bisognerebbe cioè che le testatine e i piedini siano definiti nello stile della pagina pressappoco[10] così:

```
\def\@oddhead}{\null\hfill \rightmark\ -- \leftmark}
\def\@evenhead{\rightmark\ -- \leftmark \hfill\null}
\def\@oddfoot{\null\hfill\thepage\hfill\null}
\def\@evenfoot{\null\hfill\thepage\hfill\null}
```

[10] Il "pressappoco" si riferisce al fatto che sono possibili infinite varianti.

Si noti che `\null` rappresenta una scatola vuota contro cui spinge lo spazio elastico `\hfill`; per cui la sequenza costituita dal primo e dall'ultimo lemma compare allineata con il bordo esterno della testatina, mentre i numeri di pagina sono centrati nel piedino.

È anche vero che la classe *book* è una classe generica, non direttamente utilizzabile per comporre un dizionario; sarebbe meglio usare la classe *memoir* che incorpora già molte delle estensioni e dei comandi necessari per eseguire ogni genere di personalizzazione allo stile di composizione di qualunque parte del documento che si intende comporre.

Con la classe *memoir* la personalizzazione per lo stile delle pagine interne di un dizionario si ridurrebbe alle seguenti righe, ognuna autoesplicativa:

```
\makepagestyle{dizionario}
\makeheadrule{dizionario}{\textwidth}{\normalrulethickness}
\makeevenhead{dizionario}{\bfseries\rightmark}{}{\bfseries\leftmark}
\makeoddhead{dizionario}{\bfseries\rightmark}{}{\bfseries\leftmark}
\makeevenfoot{dizionario}{}{\thepage}{}
\makeoddfoot{dizionario}{}{\thepage}{}
```

Si noti il filetto che separa la testatina dal blocco del testo; si noti che testatine e piedini sono fatti di tre parti, quella di sinistra, quella di centro e quella di destra, e a queste parti si riferiscono gli ultimi tre argomenti, alcuni lasciati vuoti, di ciascuna testatina e di ciascun piedino. Infine, in questo stile il primo lemma della pagina è indicato a sinistra e l'ultimo lemma è indicato a destra, entrambi in neretto. Ecco quindi un altro modo di personalizzare lo stile delle pagine di un dizionario.

20.7 La pagina del titolo

Non è un caso che la pagina del frontespizio non sia componibile se non con pochissimi pacchetti disponibili su CTAN; le classi per la composizione di tesi generalmente dispongono di un loro ambiente o possono usare il pacchetto di estensione *frontespizio*. L'ambiente *titlepage* della classe *book* offre uno spazio per inserire il comando `\maketitle`, ma per la verità il risultato è molto modesto. Nella prima pagina, senza usare *titlepage* si è cercato di ottenere un frontespizio accattivante da stampare sulla copertina (in brossura) di un esemplare stampato su carta. Sia questa prima pagina della copertina sia la pagina del titolo, sono modeste, ma accettabili.

Il motivo è semplice ed è chiaramente espresso da Wilson nel manuale della sua classe *memoir*: il titolo/frontespizio va disegnato apposta per ciascun documento, tenendo conto del suo contenuto, del tipo di titolo, delle immagini evocative che possono esservi associate, eccetera. Ovviamente quanto verrà eventualmente stampato sulla prima pagina della copertina è una cosa, mentre è una cosa completamente diversa quello che verrà stampato nella pagina del titolo

(solitamente la terza delle facciate composte e cucite; la prima può contenere l'occhiello, detto *half title* in inglese[11]).

Probabilmente, secondo Wilson, esistono programmi grafici interattivi più adatti per comporre la prima e la quarta di copertina nonché per comporre il frontespizio (e l'occhiello); non è nemmeno da escludere che queste pagine, e quelle che devono contenere le informazioni di carattere legale, debbano o possano essere prodotte separatamente dalle segnature che compongono il documento.

Tuttavia chi scrive ha trovato piuttosto comodo ridefinire un ambiente *titlepage* in modo che dentro ai delimitatori d'ambiente il compositore possa scrivere tutte le informazioni che ritiene necessarie; egli le introduce mediante macro come `\titolo`, `\autore`, eccetera, non dissimili da quelle che si possono usare con i nomi inglesi nel preambolo: `\title`, `\author`, `\date`; sono solo in numero molto maggiore che non quelle previste di default da LaTeX.

Il comando di apertura dell'ambiente praticamente non fa nulla, salvo verificare che si stia lavorando in una nuova pagina; il comando di chiusura fa tutto, nel senso che prende ogni informazione definita con i comandi dati dal compositore, e ogni altra informazione definita di default per collocarle in una griglia di frontespizio abbastanza elastica, ma definita una volta per tutte; esso mette anche nel verso del frontespizio (talvolta impropriamente chiamato 'colophon') tutte le informazioni di carattere legale necessarie; può, se il compositore ha specificato il numero corrente dell'opera, generare e stampare anche il *barcode* calcolando persino la cifra di controllo per il *barcode* a 13 cifre; ovviamente questo è stato fatto per una collana di volumi di una data casa editrice, dove, proprio perché si ha a che fare con una collana, i frontespizi non possono essere molto diversi l'uno dall'altro; anche gli argomenti trattati nei volumi sono sempre di carattere tecnico. Tuttavia la casa editrice, pur mantenendo lo stesso layout delle parti scritte, affida ad un grafico il compito di disegnare la prima e la quarta di copertina sovrapponendo (talvolta ricomponendo) il testo del frontespizio sopra una immagine di fondo, in modo che anche l'immagine sia di aiuto al lettore per capire (o di farsi un'idea) di che cosa venga trattato all'interno del volume.

20.8 Gli oggetti flottanti e non flottanti

20.8.1 Gli oggetti flottanti

Il pacchetto *float*, la cui documentazione si trova in `.../doc/latex/float/float.pdf`, serve per definire nuovi oggetti flottanti oltre alle tabelle e alla figure, e serve anche per dare loro uno stile, un formato adeguato al loro contenuto secondo il gusto del grafico editoriale. Naturalmente serve anche per modificare le definizioni degli oggetti flottanti già definiti.

[11] Comunque la pagina dell'occhiello non deve essere confusa con il risguardo iniziale, che è generalmente una doppia pagina di carta più pesante incollata fra la seconda di copertina e il blocco delle segnature; il risguardo all'inizio del libro e quello alla fine sono sostanzialmente due elementi strutturali usati per collegare la copertina con il blocco delle segnature; questa struttura è particolarmente importante ed è sempre presente nei libri incassati.

Per esempio può servire per specificare la collocazione della didascalia; abbiamo notato più volte che con la classe standard `book` (ma in effetti con tutte le classi standard) la didascalia è sempre collocata sotto l'oggetto flottante, mentre per le tabelle (obbligatoriamente per quelle lunghe che si sviluppano su diverse pagine) in Europa è uso mettere la didascalia prima della tabella. La stessa cosa potrebbe essere necessaria per altri stili di oggetti flottanti.

Questi potrebbero essere dei medaglioni, degli algoritmi, dei programmi, delle raccolte di formule composte in modo particolare, eccetera. È evidente allora che ogni oggetto flottante deve avere una sua numerazione, una sua coda di gestione, eventualmente un suo elenco (come l'elenco delle figure o l'elenco delle tabelle); potrebbe essere riquadrato come un medaglione, potrebbe essere delimitato da filetti; potrebbe avere lo sfondo colorato.

Il pacchetto in questione serve appunto per fare tutte queste cose, ed altre ancora. Va però notato che mentre è relativamente facile, anzi, con questo pacchetto è facilissimo definire nuovi oggetti flottanti, la parte difficile consiste nella loro gestione.

Uno dei problemi apparsi nel forum di GUIT riguardava un tipo di medaglione flottante il cui contenuto potesse svolgersi su più pagine, in modo da non interrompere per tutte le sue pagine la composizione del testo corrente; tipicamente il medaglione avrebbe dovuto avere uno sfondo colorato e ogni suo moncone avrebbe dovuto essere delimitato da filetti e da adeguate intestazioni in testa e in calce ad ogni moncone; nello stesso tempo i vari monconi, avendo deciso che ognuno non avrebbe superato la metà dell'altezza della pagina ordinaria, avrebbero dovuto uscire in pagine assolutamente consecutive, senza essere disturbati dagli oggetti flottanti di altro genere eventualmente ancora in coda.

Oggi esiste una soluzione di Agostino De Marco, membro di GUIT, che ha pubblicato il suo risultato su ArsTEXnica[12]. La soluzione molto parziale che lo scrivente aveva trovato era basata su comandi di bassissimo livello del linguaggio primitivo dei programmi di composizione e sulla programmazione avanzata eseguita mediante il comando `\whiledo` del pacchetto *`ifthen`*; tuttavia essa è evidentemente troppo complicata per essere riportata qui. Si è voluto però esporre questo caso per far capire meglio la problematica relativa agli oggetti flottanti, alla loro composizione, ma soprattutto alla loro gestione.

20.8.2 Gli oggetti non flottanti

Sembra strano che qui si aggiunga un paragrafo sugli oggetti non flottanti, ma talvolta è necessario farne uso, sia pure con estrema moderazione.

Certe figure, fotografie, riproduzioni di stemmi o altri simboli del genere, spesso vengono trattati come delle piccole figure non flottanti, alle quali, se si dovesse apporre una didascalia, sarebbe preferibile usarne una senza numero, in modo da 'essere costretti' a riferirsi ad esse in via indiretta, come per esempio 'la fotografia della pagina 125', oppure la 'fotografia qui a lato'. Questo si rende necessario per

[12] Agostino De Marco e Massimiliano Dominici hanno pubblicato un articolo su ArsTEXnica di ottobre 2008 in merito a questo problema, ma non hanno ancora pubblicato il file di estensione.

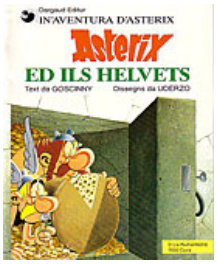
evitare che questi oggetti possano essere trattati come delle figure, e quindi gestiti in automatico dalla stessa coda che LaTeX usa per le figure; questa gestione, infatti, non mette mai una figura con un numero d'ordine superiore prima di un'altra figura con un numero d'ordine inferiore; non deve succedere che una piccola foto inserita e contornata dal testo abbia una didascalia con una numerazione più alta di un'altra fotografia di maggiori dimensioni trattenuta in memoria fino a quando LaTeX trova la pagina giusta per aggiungerla ad una pagina in uscita. Contemporaneamente la piccola figura non può essere trattenuta in memoria da una grande figura che ne ritarda la collocazione, rendendone vano l'uso.

Ecco perché sarebbe opportuno evitare gli oggetti non flottanti, ma se proprio questi oggetti devono venire usati, allora è bene che non siano trattati alla stessa stregua degli oggetti flottanti della stessa natura.

Esiste però il pacchetto *wrapfig2* che permette di gestire gli inserti contornati dal testo sia in modo fisso sia in modo più o meno flottante.

Questo pacchetto definisce due ambienti *wrapfigure* e *wraptable* simili nelle prestazioni ai normali ambienti flottanti *figure* e *table*, ma che richiedono due argomenti facoltativi e due obbligatori, il primo dei quali dice con una opportuna lettera di posizionamento se l'oggetto deve flottare o restare fisso; in entrambi i casi l'oggetto può essere collocato nel lato destro o sinistro del testo, oppure, componendo in bianca e volta, dal lato esterno o dal lato interno del testo.

La documentazione si trova in `.../doc/latex/wrapfig2/wrapfig2.pdf` che contiene anche molti utili suggerimenti in merito alla gestione di questi oggetti e descrive bene le funzionalità del pacchetto nonché le sue limitazioni.



I parametri facoltativi che l'ambiente accetta permettono di gestire con finezza lo spazio verticale da lasciare per l'inserto e di gestire l'eventuale sporgenza nel margine.

I due ambienti di *wrapfig2* hanno nomi diversi così che quando si mette una didascalia, questa è preceduta con il nome giusto e numerata con lo stesso contatore degli oggetti flottanti con lo stesso nome. Il pacchetto è compatibile con il pacchetto *float* per cui *wrapfig* riesce a gestire come inserti anche i nuovi oggetti definiti con `\newfloat`.

La sintassi (descritta per gli inserti di figure; quelli per le tabelle vengono gestiti nello stesso modo) è la seguente:

```
\begin{wrapfigure}[⟨numero di righe⟩]{⟨posizionamento⟩}[⟨sporgenza⟩]{⟨larghezza⟩}
⟨comandi per la figura⟩
\end{wrapfigure}
```

oppure

```
\begin{wrapfigure}[⟨variazione⟩]{⟨posizionamento⟩}[⟨sporgenza⟩]{⟨larghezza⟩}⋆
⟨comandi per la figura⟩
\end{wrapfigure}
```

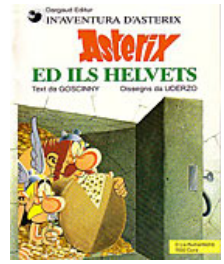
In ogni caso uno dei vantaggi di *wrapfig2* è che riesce a fare spazio per questo genere di inserti anche se devono essere contornati da diversi capoversi che,

singolarmente, non siano sufficientemente lunghi da poter contornare l'oggetto inserito.

Il ⟨*numero di righe*⟩ può essere specificato quando sia necessario regolare con finezza lo spazio verticale da destinare all'inserto. Invece, usando l'asterisco alla fine della lista degli argomenti, l'argomento ⟨*variazione*⟩ è il numero di righe che bisogna aggiungere o togliere dal numero calcolato dal software per aggiustare il rientro delle righe di testo, quando lo spazio verticale risulta leggermente diverso da quello "esatto". Il codice di ⟨*posizionamento*⟩ è una lettera minuscola o maiuscola corrispondente a `r`, `l`, `i`, `o`; le lettere minuscole specificano che l'inserto deve essere trattato come oggetto non flottante, mentre al contrario le lettere maiuscole classificano l'inserto come oggetto che può venire spostato. I codici riguardano i margini: `r`, margine destro; `l`, margine sinistro; `i`, margine interno; `o`, margine esterno. Componendo in bianca e volta conviene usare il codice `o` oppure `O`. Il parametro ⟨*sporgenza*⟩ indica di quanto l'inserto può sporgere nel margine; per impostazione predefinita questo valore è nullo. La ⟨*larghezza*⟩ rappresenta la larghezza della scatola dentro la quale il programma inserisce l'inserto; se si specifica un valore nullo, il programma usa la larghezza effettiva dell'inserto tenendo conto dell'eventuale ⟨*sporgenza*⟩. Per l'ammontare della ⟨*sporgenza*⟩ si può fare riferimento alla grandezza `\width` che è la larghezza dell'oggetto corrente da inserire. Infatti per inserire l'inserto della pagina 546 si è usato il codice:

```
\begin{wrapfigure}{O}[0.5\width]{0pt}
\includegraphics[height=8\baselineskip]{asterix-romancio}
\label{wfig:asterix}
\end{wrapfigure}
```

Volendo mettere una didascalia non numerata sotto o sopra all'inserto, basta mettere il corrispondente testo fra i ⟨*comandi per la figura*⟩ come nell'esempio a lato. Si tenga presente che per inserire una breve indicazione testuale è necessario specificare una ⟨*larghezza*⟩ non nulla; inoltre conviene specificare una forma di composizione a bandiera, perché la piccola larghezza dell'inserto non consente la composizione a pacchetto giustificata su entrambi i lati. Un inserto come questo a lato, alla fine di un paragrafo, può interferire con il titolo del paragrafo successivo e produrre difetti di composizione.



Copertina di *Asterix ed ils helvets*

20.9 Conclusioni

In questo capitolo sono stati illustrati diversi pacchetti per ottenere 'effetti speciali' nella composizione tipografica con LaTeX. Sta al compositore decidere se e come usare questi strumenti, perché, come è stato detto all'inizio, l'arte tipografica si basa su sottigliezze, non su vistosi effetti speciali.

Si è raccontato in breve come si produce un libro e come lo si disegna; ovviamente queste poche pagine non possono essere considerate nemmeno una breve introduzione alla tipografia, quindi non si raccomanderà mai abbastanza di documentarsi, non solo negli aspetti programmatori di LaTeX, ma specialmente sulla tipografia stessa, senza lasciarsi catturare dalle mode del momento, ma esercitando il proprio giudizio estetico, adeguatamente educato.

Lo scopo è quello di sviluppare una propria competenza nello scegliere le soluzioni compositive 'non acrobatiche', ma che siano effettivamente capaci di trasmettere il messaggio al lettore nel modo più chiaro possibile e senza sottoporlo a stress inutili causati da errori stilistici nella composizione.

Capitolo 21

Dove documentarsi

Invece di inserire un anonimo elenco bibliografico, che d'altronde è presente prima degli indici analitici, qui si cerca di esporre una lista ragionata di testi utili e/o necessari per approfondire la conoscenza di LaTeX e dei suoi fratelli. Le indicazioni precise di ogni riferimento sono riportate nella bibliografia, pagina 811.

21.1 La documentazione essenziale

Buona parte della documentazione relativa a LaTeX si trova già inclusa in qualunque distribuzione completa del sistema TeX.

Tutta la documentazione generale e quella dei singoli pacchetti si può leggere aprendo una "finestra comandi", che prende il nome di "Prompt dei comandi" nelle macchine Windows, e "terminale" o "console" o "xterm" nei sistemi UNIX: Mac e Linux. Aperta una di queste finestre basta dare il comando

`texdoc` ⟨*nome del documento*⟩

e poi premere il tasto Invio o Enter o ↩ . Il ⟨*nome del documento*⟩ è generalmente coincidente con il nome del pacchetto di cui si vuole leggere la documentazione o al nome della guida che si vuole leggere.

Qui elenco solo la documentazione essenziale:

usrguide generica guida iniziale per ogni utente di LaTeX.

fntguide generica guida per la manipolazione dei font con la descrizione dei comandi specifici sia per il modo testo sia per il modo matematico.

clsguide generica guida per la descrizione delle funzionalità delle classi e dei pacchetti, ma anche guida per chi si vuole scrivere delle classi o dei pacchetti personalizzati.

grfguide guida per usare correttamente i comandi per manipolare gli oggetti grafici, sostanzialmente legata all'uso dei pacchetti *graphics*, *graphicx* e *color*.

cnfguide per gestire e usare bene i file di configurazione.

cmfonts in cui la prima parte della guida indica quali siano i font con le codifiche standard della collezione Computer Modern, e poi come siano fatti i file di descrizione dei font che LaTeX usa per selezionare il font giusto al momento giusto quando si compone un documento mediante quella collezione di font.

cyrguide descrive i font cirillici e i file che consentono di comporre in russo e in ucraino.

encguide descrive le codifiche dei font e come sceglierle quando si usa LaTeX.

inputenc descrive le codifiche di entrata usabili con i diversi sistemi operativi. Oggi diventata di minore importanza, perché tutti i motori di composizione moderni si aspettano di dover gestire file sorgente che contengono soo font codificati in UTF-8.

makeindx descrive l'estensione per creare convenientemente un file per l'indice analitico e definisce il comando `\printindex`.

modguide espone i problemi che si possono incontrare se si modificano i file della distribuzione del sistema TeX, in particolare quelli relativi all'uso e alle estensioni di LaTeX, e suggerisce come poter fare le modifiche funzionali senza modificare davvero i file.

Queste guide dovrebbero essere le prime ad essere lette e le prime a venire comprese; alcune sono difficilotte per chi si trova alle prime armi. Quindi nel seguito verrano elencate anche altre guide più "onnicomprensive", cioè meno legate ad uno specifico tema (le classi, i font, la grafica, eccetera). Ma queste guide devono essere sempre tenute presenti, cioè le si deve saper consultare senza esitazioni; non è che si debbano imparare e memoria, ci mancherebbe altro, ma sarebbe opportuno saperle consultare con il programma texdoc sapendo aprire al primo colpo la guida giusta, perché si conosce già la tipologia di ciò che si desidera controllare.

Si tenga presente che anche il forum di GuIt ha una sezione Documentazione che contiene guide generali e guide tematiche tutte scritte in italiano. Se ne parlerà anche più avanti.

21.2 Documentazione sulla tipografia

Questa sezione avrebbe l'ambizione di fornire qualche idea sulla tipografia, le consuetudini tipografiche e il book design, senza limitarsi alla consuetudini italiane. Per ovvi motivi questa sezione è abbondantemente incompleta e la scelta dei riferimenti è frutto di un compromesso.

Una breve guida sulle consuetudini della tipografia italiana è contenuta nell'articolo di Gustavo Cevolani "Norme tipografiche", [16].

Questo articolo ha senz'altro il pregio della concisione, una decina di pagine, ma è piuttosto completo e indica anche come le varie consuetudini tipografiche italiane possono essere rese con i comandi e gli ambienti LaTeX.

Benché abbia i suoi anni, il manuale di Roberto Lesina [41] raccoglie in modo ben strutturato un certo numero di consuetudini tipografiche e raccomandazioni di vario genere, che molti scrittori tecnico–scientifici considerano assai valide.

Anche la documentazione accessoria della classe *memoir* [73] contiene non poche pagine di iniziazione alla tipografia e al book design. Anche se il testo è in inglese e si riferisce principalmente alla tipografia inglese, non mancano riferimenti alla tipografia europea continentale di varie nazioni e di vari secoli.

Precedentemente questa documentazione era contenuta nel manuale stesso di *memoir* [74], ma nel 2009 il curatore della documentazione ha ritenuto che il testo fosse diventato troppo lungo e ha separato questo documento dal manuale vero e proprio; viene installato insieme a TeX Live; se non fosse presente nella propria distribuzione bisogna scaricarlo da `http://www.ctan.org/tex-archive/info/memdesign/`.

Il testo elettronico di Peter Flynn [25] è sostanzialmente una guida all'uso di LaTeX, ma l'autore è professore di Tipografia all'University College di Cork, in Irlanda. Fu lui il presidente e organizzatore della Conferenza Internazionale di Cork, dove furono gettate le basi per l'encoding esteso T1. Data la professione dell'autore, il testo è particolarmente curato nella sua forma grafica e compositiva; è vero, ci sono delle convenzioni insolite per la tipografia italiana; per esempio i richiami delle note seguono la punteggiatura, mentre nella maggior parte dei libri italiani i richiami di nota precedono la punteggiatura. Tuttavia il tipo di composizione è piuttosto curata, certamente può costituire un esempio da seguire.

L'Imprimerie Nationale francese, l'equivalente del nostro Poligrafico dello Stato, ha pubblicato un interessante libretto [48]. Si tratta di una specie di glossario dove sono elencate in ordine alfabetico le varie parole che comportano certe consuetudini tipografiche e viene specificato come quelle consuetudini siano applicate presso questo autorevolissimo ente.

Tipica della tradizione tipografica francese è la spaziatura uniforme fra le parole indipendentemente da quali segni di interpunzione le separino; questo effetto con LaTeX si ottiene con la dichiarazione `\frenchspacing`. Vale la pena di segnalare che questo tipo di spaziatura è quello di default quando viene composta la bibliografia; in questo capoverso si è appunto specificata la spaziatura francese.

Bisogna osservare che le tradizioni tipografiche francesi sotto certi aspetti sono simili a quelle italiane, e sotto certi altri sono molto divergenti: non è solo una questione di lingua, ma si tratta proprio di consuetudini, una volta, molti decenni fa, usate anche in Italia. Per esempio i francesi inseriscono degli spazi prima di tutti i segni di interpunzione 'alti', dai due punti ai punti esclamativo e interrogativo; in questo libretto, anche se composto con cura, a me pare assai sgradevole il fatto che lo spazio che precede questi segni sia piuttosto grande e suscettibile di essere ulteriormente allargato per giustificare le righe; forse la sgradevole sensazione dipende dalla mancanza di abitudine. Questa sensazione

mi si ripresenta quando vedo i numeri romani usati come ordinali ma con la desinenza ad apice, come in 'Lille est le siège de la II^{e} région militaire'; in italiano sarebbe certamente scorretto mettere ad apice di un numero romano la desinenza dell'aggettivo ordinale.

Bisogna però ammettere che questo è uno dei pochi manuali di composizione che tratta anche gli aspetti che non vengono mai trattati dagli autori più famosi; per esempio la composizione della matematica, l'uso delle unità di misura, la scrittura della chimica, anche quella organica con le sue formule di struttura. Anche in matematica la tipografia francese si distingue con particolarità tutte sue, tuttavia le indicazioni fornite in questo libro sono certamente una buona guida anche per gli italiani.

Abbastanza documentato e facile anche per i neofiti è il libro di Michael Mitchel e Susan Wightman [43]. Esso copre tutti gli aspetti della produzione di un libro, e dà indicazioni assai utili a chi desidera fare o fa di professione il book designer, sia pure in un contesto britannico. In realtà, fuori dal Regno Unito cambiano i dettagli, ma i problemi da affrontare per la produzione di un libro sono praticamente gli stessi in ogni nazione. Un dilettante legge volentieri questo manuale, perché non è formato da una serie di prescrizioni, ma è un susseguirsi di descrizioni dei vari aspetti, inclusi quelli tipografici e compositivi, naturalmente, ma è molto istruttivo. Gli autori non fanno un mistero di avere usato un prodotto professionale e commerciale per la *mise en page*, come direbbero i francesi, ma si sforzano di dare indicazioni per non vincolare il lettore allo stesso programma di impaginazione. Molti dei problemi da risolvere a mano con gli impaginatori professionali sono risolti automaticamente con LaTeX, ma ce ne sono altri che con LaTeX sarebbe molto difficile affrontare e risolvere. È utile domandarsi via via: "Questo con LaTeX come potrebbe essere fatto?".

Fra i grandi book designer di questo secolo compare anche Robert Bringhurst; egli ha scritto un piacevole libro [11] dal titolo significativo: *The Elements of Typographic Style*, dove egli descrive i suoi principi tipografici e di book and page design; è incredibile come riesca a legare le proporzioni delle note musicali delle scale diatonica e cromatica con quelle delle figure geometriche e con quelle da usarsi in tipografia. Nelle sue mani quei principi per la scelta delle proporzioni diventano davvero musica, per gli occhi, invece che per le orecchie, ma con quella sobrietà che consente alla persona di gusto di apprezzarne le finezze discrete e sobrie, che però non si impongano fino a distrarre il lettore. Leggendo questo libro e comprendendo appieno i principi tipografici di Bringhurst, si capisce che il book design di qualità non è un gioco da apprendisti.

Gli ultimi riferimenti sono testi in inglese; è un peccato che non esistano o siano difficili da trovare opere simili in italiano[1]; d'altra parte è anche comprensibile che la pubblicazione di opere di questo genere in italiano per lettori italiani non sarebbe remunerativo, mentre lo è certamente se sono pubblicate in inglese per i lettori che usano l'inglese come prima o seconda lingua.

[1] Il testo di Bringhurst esiste anche in italiano, ma oggi viene stampato come *book on demand*.

Tra le opere italiane, però, si può citare il manuale di Giorgio Fioravanti [24] dal titolo: *Il manuale del grafico — Guida alla progettazione grafica e all'impaginazione del prodotto editoriale.*

Come dice il titolo, questo libro è più rivolto agli aspetti grafici, piuttosto che a quelli compositivi; penso che si tratti di una buona lettura, ma certamente non ha il respiro dell'opera di Bringhurst o la copertura di argomenti di Mitchel e Wightman. Tuttavia merita notare la composizione moderna a gabbia su tre colonne, dove spesso due delle colonne sono unite assieme, così che le pagine sono mosse e attraenti; in questa apparente disomogeneità la struttura grafica si adegua al contenuto, senza però attrarre l'attenzione su di sé distraendo il lettore dalla comprensione del testo.

Si può citare anche il libro di Fabrizio Serra [58]; l'autore, oltre ad essere proprietario e amministratore di un certo numero di aziende tipografiche, è stato anche professore di Composizione della stampa presso la Scuola diretta ai fini speciali in Scienza ed Arti della Stampa, ora divenuto il corso di laurea in Progetto Grafico e Virtuale. Le sue aziende si occupano specialmente, ma non esclusivamente, di stampa di libri, di collane e di periodici nell'ambito delle discipline umanistiche. Il libro di norme che Fabrizio Serra ha pubblicato si riferisce prevalentemente alla stampa di documenti testuali, con numerosi inserti iconografici, tabellari, eccetera, ma in genere senza espressioni matematiche. Il libro contiene una prefazione, una postfazione e una appendice scritte da personaggi importanti nell'ambito della tipografia e del *book design*; basta citare fra tutti Jan Tschichold, famosissimo book designer svizzero.

21.3 Documentazione su LaTeX

La fonte iniziale di ogni sapere a proposito di LaTeX standard è il manuale di Leslie Lamport [39]. L'appendice C di questo testo è stata tradotta e commentata in italiano, [6]; assomiglia al capitolo 29 di questa Introduzione, ma ha una sua autonomia.

Per leggere un testo introduttivo più breve del manuale di Lamport, si può ricorrere al testo [4], inizialmente scritto in tedesco, poi tradotto in molte lingue, fra le quali l'italiano; la traduzione italiana è stata curata da un certo numero di membri del GuIt.

In italiano è disponibile il testo di Lorenzo Pantieri e Tommaso Gordini [55] dal titolo: *L'Arte di scrivere con LaTeX*. È un'ottima guida, e viene aggiornata abbastanza spesso; tipograficamente parlando è molto più bella di questa Introduzione; l'autore, infatti, si è servito del pacchetto di estensione `ClassicThesis`, che modifica le prestazioni delle classi della serie KOMA Script, di cui si è già parlato, producendo un layout assai elegante e composto con font scelti con cura; al di là del contenuto, interessantissimo e ben scritto, anche lo stile tipografico può costituire un buon modello a cui ispirarsi. Invece in questa Introduzione si sono usati la classe standard `book` e i pacchetti standard in modo da mostrare quello che LaTeX può fare da solo. Nel testo di Pantieri e Gordini, invece, è stato

cambiato praticamente tutto; il risultato è eccellente, ma per i neofiti potrebbe risultare difficile sfruttare le possibilità di `ClassicThesis` per ottenere le stesse bellissime pagine.

Lorenzo Pantieri ha scritto anche un'altra breve guida, *LaTeX per l'impaziente*, che potrebbe sostituire la guida [4], ripubblicata e aggiornata nel 2019. La guida di Pantieri [53] contiene molti riferimenti ai pacchetti di estensione, in modo da indirizzare subito il lettore impaziente verso le soluzioni compositive migliori; ovviamente espone anche le parti essenziali di LaTeX, ma non tratta quasi per niente gli argomenti che esulano dalla composizione del testo con il criterio di "quello che vedi è quello che intendi dire", tipico di LaTeX. Il testo è composto con la classe `book` con l'estensione `layaureo`. Il lettore curioso può vedere subito che effetto fa usare questo pacchetto, a confronto con questa Introduzione dove, invece, l'aspetto grafico della pagina è quella di default della classe.

Lorenzo Pantieri ha anche scritto un altro testo intitolato *LaTeXpedia*, [54]. Questa guida raccoglie ciò che Pantieri ha scritto in merito a LaTeX e ai pacchetti con i quali ha lavorato nell'arco di anni. L'impaginazione è identica a quella de *l'Arte*, il contenuto è molto più vasto e comprende anche argomenti molto avanzati.

Invece, per andare su un testo più completo anche della Guida di Lamport ci si può riferire all'eccellente libro di Kopka a Daly [37].

Ma il libro più completo (e pesante, non nel prezzo, ma nel numero di pagine), il libro dei libri su LaTeX, è certamente il testo *The LaTeX companion* [44]. Si tratta di un libro di quasi 1100 pagine dove c'è tutto lo scibile concernente LaTeX 2_ε e i più importanti pacchetti. è stato ripubblicato e aggiornato nel 2023.

21.4 Documentazione sulla grafica

È stata pubblicata anche la seconda edizione del volume *The LaTeX graphics companion* [45]. Questo libro è utilissimo per apprendere come manipolare le immagini per poterle inserire al meglio all'interno di un documento composto con LaTeX. Sono citati e descritti in dettaglio molti dei programmi appena menzionati in questo testo, ma, quello che più importa, essi sono commentati e confrontati per esplorarne pregi e difetti o per indicare usi alternativi a quelli che ci si potrebbe aspettare[2].

Per quel che riguarda la creazione di grafica che rappresenta informazione quantitativa, si indica il libro di Edward Tufte, *The visual display of quantitative information*. Oltre ad essere un bellissimo libro in sé, esso mostra una quantità di rappresentazioni grafiche belle e brutte, tutte commentate; per quelle brutte i commenti indicano proprio perché sono mal riuscite. C'è molto da imparare, anche per chi si crede un abile creatore di diagrammi, cartogrammi, istogrammi, eccetera. Ciò non toglie che non si possano tenere in conto anche le raccomandazioni della

[2]Tutti i libri della Addison Wesley sono acquistabili tramite il sito dell'associazione mondiale degli utenti di TeX con notevoli sconti; si suggerisce di esplorare il sito `http://www.tug.org/books/`.

norma UNI 2949, [64], che, guarda caso, sono coerenti con la maggior parte delle indicazioni di Tufte, anche se hanno un aspetto più tecnico dei bei diagrammi mostrati da Tufte nel suo libro.

21.5 Documentazione sui singoli pacchetti

La documentazione sui singoli pacchetti di estensione fa generalmente parte della distribuzione stessa del pacchetto; ogni volta che si scarica dalla rete un pacchetto e questo viene installato nella struttura standard di cartelle del sistema TEX, la documentazione va a finire in una cartella, che si chiama come il pacchetto, nel ramo dell'albero che comincia con `.../doc`; può poi finire nel ramo `latex` oppure `tex` oppure `generic` a seconda che l'uso del pacchetto sia specifico di LATEX, di TEX, oppure sia polivalente.

Queste cartelle sono le prime a dover essere esaminate quando si cerca della documentazione specifica; non lo si raccomanda mai abbastanza, tanto che molti utenti si rivolgono in rete, alla prima mailing list che capita, per avere delucidazioni su cose che sono spiegate chiarissimamente nella documentazione che arriva insieme al sistema TEX e che quindi è già installata nello stesso calcolatore dove l'utente si è trovato di fronte ad un problema. Per carità un po' di aiuto non lo si nega a nessuno, ma quando si chiede aiuto ad una mailing list competente, la 'netiquette' impone prima di tutto di aver fatto il possibile per documentarsi autonomamente.

Per i problemi relativi al sistema TEX è certamente competente il forum `comp.text.tex` in `groups.google.com`.

Anche il Gruppo degli utilizzatori Italiani di TEX ha un forum all'indirizzo `https://www.guitex.org/forum/` ed è meglio rivolgersi a questi gruppi, invece di scrivere alla mailing list dello shell editor che si sta usando, dimostrando quindi di non aver capito la distinzione fra lo shell editor e il sistema TEX.

Ma una cartella che bisogna avere sempre presente è `.../doc/latex/base/` perché quella cartella contiene alcune rapide guide dal titolo *xxx*`guide.pdf`, dove le tre lettere *xxx* indicano un po' stenograficamente di che cosa si tratti: *usr* riguarda il generico utente, *cls* riguarda le classi, eccetera; anche se sono abbreviazioni inglesi non è difficile decifrarle. Quelle guide sono già state elencate nel preambolo di questo capitolo.

Un problema per gli utenti italiani è costituito dal fatto che quasi tutta la documentazione è scritta in inglese; capisco che non tutti sono tenuti a conoscere l'inglese come la propria lingua madre, tuttavia se si usa un PC o un laptop, se si naviga in rete, una conoscenza di base dell'inglese è sicuramente presente; questa conoscenza di base dell'inglese scritto dovrebbe essere sufficiente.

21.6 Documentazione su TEX

Se si arriva a scrivere dei file di classe o dei pacchetti di estensione viene voglia di conoscere meglio il linguaggio TEX nativo; il libro dei libri è il TEXbook [35], scritto direttamente dal padre del sistema TEX. Questo libro è la fonte più autorevole su TEX; se si è in vena di spese, esiste anche l'*opera omnia* di Knuth relativa alla tipografia digitale pubblicata come "Millenium Edition" [36]. Essa contiene, in un elegante cofanetto cartonato, tutti e cinque i volumi del sistema TEX: Volume A, *The TEXbook*; Volume B, *TEX: The Program*; Volume C, *The* METAFONT*book*; Volume D, METAFONT*: The Program*; Volume E, *Computer Modern Typefaces*. Tutti e cinque i volumi sono incassati (copertina rigida) e, a seconda dell'uso che se ne fa, sono molto meglio che non procurarsi separatamente le varie parti, per giunta brossurate o con legature a spirale.

Conviene visitare il sito di Enrico Gregorio: `http://profs.sci.univr.it/~gregorio/egtex.html`. Ci sono diversi documenti da scaricare, da quello sugli Orrori, a quello sugli Esempi. Quello che sembra più interessante, oltre alle altre Guide sui pacchetti più comuni, è il testo PDF *Appunti di programmazione in TEX e LATEX*; si tratta della seconda edizione aggiornata nel 2009 e contiene diverse note che descrivono molte macro sia scritte in linguaggio TEX, sia in linguaggio LATEX 2ε. Si tratta di cose molto istruttive, alcune semplici, altre decisamente più complesse; ma come testo didattico, secondo chi scrive, è un ottimo esempio che consente all'apprendista di arrivare per gradi e secondo le sue necessità a livelli alti di programmazione, ottimi per scrivere macro complesse, così come per scrivere file di classe o di estensione. Non si dimentichino i molti articoli che Enrico Gregorio ha scritto epr ArsTEXnica; sono tutti scaricabili dalla sezione ArsTeXnica del forum del GUIT: `www.guitex.org`.

21.7 Documentazione sui simboli di LATEX

LATEX consente di inserire nei documenti una miriade di simboli che sono disponibili con certi pacchetti oppure con certi font; esiste un documento in rete, predisposto da Scott Pakin e tenuto costantemente aggiornato, che contiene tutti i simboli gestibili con LATEX, [50]. L'indirizzo in rete rimane sempre lo stesso anche dopo ogni aggiornamento. Per altro con il sistema TEX si può leggere direttamente com `texdoc symbols-a4`.

21.8 Documentazione sulla composizione della matematica

In una lettera privata a un suo corrispondente Jean-Pierre Serre scrisse:

> Mi colpisce il fatto che la scrittura matematica sia simile alla scrittura di una lingua. Per essere capito devi seguire certe regole grammaticali. Tuttavia, nel nostro caso, nessuno si è preso la briga di scrivere una

> grammatica; noi l'afferriamo come un bimbo lo fa dai suoi genitori, per imitazione degli altri. Alcuni matematici hanno un buon orecchio; altri no (e alcuni preferiscono usare espressioni gergali come, per esempio, "iff"). Così è la vita.[3]

Tuttavia non è proprio completamente esatta questa affermazione; alcune regole esistono, almeno per certi stili di scrittura matematica.

Le norme ISO per la composizione della matematica scritta per i fisici e i cultori delle scienze sperimentali e tecnologiche (la matematica delle grandezze) si trovano tradotte in italiano nel documento [23] pubblicato dall'Ente di Unificazione Italiano. Fra le guide tematiche del GUIT, compare anche *Regole e consigli per comporre la matematica delle scienze sperimentali*, [7]. Si ritiene che sia il minimo necessario per comporre bene la matematica delle scienze che usano grandezze. Alcune nozioni sono utili anche per chi scrive di matematica pura, ma essenzialmente il testo è rivolto ai fisici, agli ingegneri, ai chimici, e a tutti coloro che usano la matematica delle grandezze.

Vale la pena di consultare anche le indicazioni fornite dalla Unione Internazionale di Fisica Pura e Applicata riportate per esempio nell'appendice F del testo [72].

Molto interessante e completo è il documento elettronico [49] pubblicato dal National Institute for Standardization and Technology; esso è estremamente dettagliato e vale la pena di disporne e di ricorrevi spessissimo; è anche molto aggiornato sul sistema SI; ovviamente si rivolge ai lettori statunitensi, quindi dove discute del separatore decimale indica il punto invece della virgola; usa anche l'ortografia americana invece di quella inglese per rispettare usi e costumi consolidati negli USA, per giunta fissati per legge da altre norme. La bibliografia è molto estesa; fra le appendici ci sono elenchi di fattori di conversione da unità "strane" a unità SI che risultano difficilissimi da reperire altrove; alcune di queste unità "strane", sono tanto inconsuete che se non fossero elencate in quell'appendice qui in Europa non si sarebbero mai potute incontrare. Le norme in questione, complete di check list per gli autori, sono destinate ai vari ricercatori di quell'Istituto affinché scrivano i loro testi tecnici in modo irreprensibile. Fatte le debite piccolissime tare (come il punto decimale e il plurale dei nomi che indicano le unità di misura; in italiano quei nomi — quelli derivanti dai nomi di scienziati e quelli che finiscono in consonante — sono tutti invarianti al plurale) le indicazioni presenti in questa guida sono validissime per qualunque scrittore tecnico.

L'American Mathematical Society ha anche pubblicato due libretti su questo argomento; il primo [61] serve per dare adeguate istruzioni agli autori che le inviano lavori matematici da pubblicare; il secondo [38], invece, oltre a dedicare

[3] It strikes me that mathematical writing is similar to using a language. To be understood you have to follow some grammatical rules. However, in our case, nobody has taken the trouble of writing down the grammar; we get it as a baby does from its parents, by imitation of others. Some mathematicians have a good ear; some not (and some prefer the slangy expressions such as "iff"). That's life.

un adeguato spazio alla composizione della matematica con il calcolatore, serve anche per consigliare il lettore come scrivere di matematica, cioè come usare la prosa che può interessare un matematico.

Come appare chiaro dal titolo, i primi riferimenti si riferiscono esplicitamente alla matematica scritta dai fisici e dagli scienziati sperimentali, mentre i secondi si riferiscono alla matematica dei matematici. Come è stato giustamente osservato, sarebbe opportuno che fisici, matematici, ingegneri e scienziati sperimentali scrivessero la matematica nello stesso modo: non c'è dubbio. Ma non conoscendo se il lettore sia un matematico, o un fisico, o un ingegnere, o un economista, o un giurista, eccetera, allora...

In ogni caso si rinvia al capitolo 24 una lista di simboli e di notazioni tratta dalle norme o dalle pubblicazioni suddette.

Può essere utile il manualetto preparato da Michael Downes [20], della American Mathematical Society, ripercorre un po' quanto si è già detto a proposito delle estensioni fornite dal pacchetto *amsmath*, ma scende in maggiori dettagli e fornisce utilissimi commenti e indicazioni di ulteriori pacchetti di estensione della matematica che qui non sono stati nemmeno nominati.

Unendo "l'utile al dilettevole" si segnala il pacchetto *cool* [59]; esso propone all'utente un certo numero di estensioni che consentono composizioni matematiche ben fatte e ben documentate. Leggibile direttamente con `texdoc cool`.

Esso può essere utilissimo per comporre la matematica sia con lo stile dei matematici, sia con quello dei fisici e ingegneri; esso consente di usare una moltitudine di comandi ognuno dei quali realizza la corretta scrittura di ogni possibile piccola o grande operazione matematica, con la garanzia che essa sarà tradotta nel file di uscita in modo corretto; ma esso consente anche una maggiore leggibilità del file sorgente in quanto i nomi degli operatori o delle operazioni sono scritti per disteso o comunque sono poco abbreviati; il pacchetto contiene definizioni anche per gli operatori più specializzati e che non sono listati nel capitolo 24. Se ne consiglia l'uso a tutti gli utenti di LaTeX che abbiano necessità di comporre testi con una notevole dose di matematica, in particolare se questa è molto avanzata.

Si consiglia vivamente la lettura del documento di Herbert Voß, *Math mode*, [71], che espone tutti i modi possibili per comporre la matematica sia con plain TeX, sia con il mark-up LaTeX, senza o con le estensioni fornite principalmente dal pacchetto *amsmath*, e pacchetti annessi, oltre che con un'altra ventina di pacchetti specializzati per la composizione della matematica. Esamina i problemi di composizione per categorie e spessissimo mostra degli esempi, completi di codice, anche per realizzare strutture compositive matematiche per situazioni particolari. Sono 135 pagine densissime di informazioni preziose e costituiscono una vera miniera per trovare soluzioni o raccogliere idee nuove.

Benché si trovi nella cartella `CTAN/obsolete/info/math/voss/mathmode` dell'archivio CTAN, il file è stato aggiornato nel 2014; viene (apparentemente) ancora distribuito con l'intallazione MiKTeX, ma chiunque ne può scaricarne il file `mahmode.pdf` e leggerselo tranquillamente sul proprio elaboratore. A suo

tempo era distribuito anche con TEX Live, ma evidentemente gli ampliamenti di questa installazione ne hanno reso superflua la distribuzione.

Per quel che riguarda la *vexata quaestio* della punteggiatura esterna delle formule in display, oltre a richiamare quanto annotato verso la fine del capitolo 14 si suggerisce qui l'esame dei fascicoletti di istruzioni per gli autori messi a disposizione su CTAN e nelle apposite cartelle delle varie distribuzioni del sistema TEX. Si ricorda il già citato fascicoletto della Elsevier [22] e l'analogo fascicoletto predisposto dalla Kluwer [34]. Queste case editrici specializzate nelle pubblicazioni di tipo accademico e scientifico non usano la punteggiatura esterna per le espressioni matematiche in display.

La bibliografia di questo testo contiene diversi riferimenti a libri o manuali che trattano di punteggiatura; sono tutti scritti da linguisti quindi non si occupano della punteggiatura in matematica. Tuttavia c'è il testo [57] che, secondo chi scrive, ha una visione più ampia degli altri sul significato di "segno di punteggiatura"; infatti oltre ai soliti segni convenzionalmente considerati per punteggiare un testo, ci sono altri segni, compresi gli spazi orizzontali e verticali che possono essere usati per una marcatura logico-sintattica del testo. Si tratta di un punto di vista abbastanza insolito, ma utile per comprendere e valutare meglio anche la questione della punteggiatura esterna e, parzialmente, di quella interna, realizzata mediante spazi nel campo della matematica.

21.9 L'archiviazione dei documenti

L'argomento relativo all'archiviazione è ancora abbastanza nuovo e c'è poca documentazione disponibile. L'unico testo che sembra essere stato pubblicato è in vendita solo accedendovi dalla rete, [21].

In compenso nel sito `http://www.pdfa.org/doku.php?id=start:en` si trova una quantità di informazioni incredibile, che vanno dai documenti tecnici, alle *FAQ*, le *Domande frequenti*, dove si trovano una quantità di informazioni spicciole su argomenti precisi e delimitati.

Il sito viene continuamente aggiornato e vale la pena di accedervi regolarmente, specialmente se la produzione di documenti archiviabili è una parte importante del proprio lavoro.

III

Volume

Capitolo 22

Il formato PDF archiviabile

In una guida come questa non può mancare un capitoletto sul formato PDF archiviabile. Questo formato soddisfa alla norma ISO 19005 del 2005 (e successive modificazioni) che stabilisce particolari restrizioni o specifiche al formato.

Come questa Guida che nasce essenzialmente come libro elettronico (*e-book*), così una miriade di altri documenti nasce sotto forma di documenti elettronici, che talvolta vengono stampati, ma sempre vengono archiviati. Come sarà possibile leggere questi documenti di archivio di qui a cinquanta anni? Ecco, questo è il motivo per il quale l'ISO ha provveduto a emettere le norme per l'archiviazione dei documenti elettronici.

Sostanzialmente il formato prescelto dall'ISO è il PDF (*Portable Document Format*) che già in partenza era nato come uno strumento relativamente aperto ma comunque indipendente dalla piattaforma sulla quale ogni documento era stato composto. La Adobe, che aveva sviluppato questo formato, aveva reso pubbliche le specifiche in modo che chiunque potesse scrivere programmi di conversione, di composizione o di visualizzazione per questo formato.

Gli utenti di TEX ne sono perfettamente consapevoli, visto che usano i programmi di composizione del sistema TEX nato in modo indipendente dalla Adobe; usano anche ghostscript, nato per leggere e stampare i documenti scritti in formato PostScript, e successivamente esteso per leggere e scrivere (convertire) documenti in formato PDF. La collezione dei programmi del sistema TEX contiene anche il programma dvipdfm per convertire direttamente un file in formato DVI in un file corrispondente in formato PDF; lo stesso avviene per i vari altri programmi "fratelli" di pdftex: xetex, aleph, context, luatex, eccetera. Nessuno di questi programmi è nato presso la Adobe, ma è stato possibile predisporli grazie al fatto che le specifiche del formato sono a disposizione di chiunque. Inoltre quasi tutti i word processor consentono di esportare il loro documenti nel formato PDF.

Era naturale che con queste premesse l'ISO si rivolgesse al formato PDF come base per il formato archiviabile. Essa però ha imposto che il documento archiviabile sia scritto con il linguaggio PDF nella versione 1.4 (con le successive

versioni della norma anche livelli del linguaggio PDF più elevati) e che sia dotato di altri dati che consentano il reperimento dell'informazione utile con i sistemi archivistici più avanzati; per esempio, esso deve contenere dei *metadati* che consentano come minimo di ricercare il documento sulla base del suo titolo e/o dei suoi autori; ancor meglio: questi *metadati* possono contenere anche le parole chiave per eseguire ricerche più mirate.

L'ISO impone anche che il documento sia indipendente dalle risorse dell'elaboratore sul quale viene eventualmente letto dopo molti anni dalla sua archiviazione; ovviamente l'elaboratore dovrà disporre di un lettore di documenti PDF, ma non necessariamente uno dei lettori esistenti oggi. Non basta: il documento deve essere autosufficiente anche per quel che riguarda i font con cui è composto, e deve completamente integrare ogni immagine della quale sia anche ben chiaro il modello di colore usato.

In sostanza i requisiti minimi richiesti ad un documento archiviabile, oltre ai *metadati* per le ricerche di archivio, sono che il documento sia leggibile esattamente nella stessa forma in cui era stato composto. Questi requisiti minimi configurano la specifica PDF/A-1b, meno restrittiva della specifica PDF/A-1a, la quale invece richiede che il file PDF sia anche un *tagged PDF* affinché sia estraibile anche la struttura logica del documento, indipendentemente dal fatto che questo contenga gli hyperlink interni per la lettura dinamica sullo schermo. Le versioni successive della norma ISO relativa all'archiviabilità prevedono anche altre sottonorme, l'ultima delle quali, ancora in fase sperimentale, prevede delle ulteriori specifiche per l'accessibilità ai disabili. Qui ci occuperemo solo della norma PDF/A-1b, anche perché allo stato attuale la creazione dei file di tipo "tagged PDF" non è ancora completamente eseguibile con i programmi di composizione del sistema TEX basati sul mark-up LATEX: pdfLATEX, LuaLATEXe XeLATEX; si stanno sviluppando le necessarie funzionalità e per ora si dispone del pacchetto sperimentale `tagpdf`, la cui documentazione è leggibile con `texdoc tagpdf`.

Ovviamente la Adobe ha provveduto con il suo programma Acrobat Professional, dalla versione 7.0 in poi, a rendere disponibile un certo numero di opzioni per trasformare certi file nati in formato PDF nei requisiti richiesti dalle norme ISO, sia per la variante PDF/A-1a sia per quella PDF/A-1b e di altri livelli. Tra le opzioni c'è anche la possibilità di *verificare* se un file PDF prodotto esternamente sia già conforme a una delle specifiche ISO di diversi livelli; eventualmente ne consente la correzione, in ogni caso ne segnala gli errori e le manchevolezze.

In commercio esistono diversi altri programmi di conversione e di verifica ma, a conoscenza dello scrivente e fino ad oggi (fine 2016), nessuno di quei programmi è disponibile gratuitamente, salvo, ovviamente, i programmi del sistema TEX. Infatti questo sistema contiene ora (dicembre 2018) i mezzi necessari per produrre i file PDF/A-1b, e secondo altre norme meno frequenti per l'archiviazione; esiste anche il programma ghostscript (dalla versione 8.60 in poi) che afferma di poter eseguire la conversione, in modo diverso, ma conforme alle specifiche. Quello che il sistema TEX non contiene ancora è un programma di verifica della conformità, e l'utente non può fidarsi del fatto che, se l'esecuzione di pdflatex o di ghostscript

è andata a buon fine, allora siano anche rispettate tutte le specifiche della norma ISO. A tutt'oggi (2018) la verifica più affidabile è quella eseguita con il modulo Preflight del programma Adobe Acrobat Pro XI, o con il verificatore gratuito VeraPDF reso disponibile in modo stabile dal 2017; esso è stato creato da un gruppo di lavoro incaricato dall'Unione Europea. Oggi (2024) VeraPDF consente anche di eseguire una verifica on line.

La produzione di un documento archiviabile non fa parte in senso stretto dell'Arte della Composizione Tipografica; tuttavia se un documento è stato prodotto con ogni cura e deve essere archiviato, è importante che il compositore segua anche quelle norme e quelle procedure che permettono di rispettare le norme ISO.

22.1 Preliminari

Per capire bene i vincoli che le norme impongono ai file archiviabili conviene subito fare un elenco; i dettagli, anche operativi, verrano esposti nei paragrafi successivi.

1. Un file archiviabile deve avere il formato PDF.
2. Il livello del linguaggio PDF deve essere 1.4; revisioni della norma ISO successive alla prima emanazione ammettono anche varianti che possono usare fino al livello 1.7, ma qui ci riferiremo solamente al livello 1.4.
3. Ogni file archiviabile deve essere dotato di metadati in chiaro, non criptati né compressi.
4. Questi metadati devono essere conformi alle norme stabilite inizialmente in una conferenza detta "di Dublino", che costituiscono il Dublin Core Metadata Element Set (DCMES), riconosciuto dalla norma ISO 15836-1:2017.
5. I font con cui è composto il documento devono essere vettoriali, preferibilmente con codifica UNICODE, comunque la norma ammette anche i font Type 1 e TrueType, purché nel documento da archiviare sia integrato il file di codifica contenente le denominazioni UNICODE.
6. Ogni glifo usato nel documento deve essere corredato delle informazioni metriche e, in particolare, la larghezza non deve essere mai nulla.
7. Le immagini non devono contenere trasparenze, cioè aree che lasciano trasparire il colore sottostante.
8. Ogni immagine deve avere un ben preciso profilo di colore, uguale per tutte le immagini di un dato documento.
9. In ogni caso i file che contengono le specifiche dei profili di colore devono essere integrati nel documento da archiviare.
10. Non sono ammessi contenuti multimediali con le norme PDF/A-1a e PDF/A-1b.
11. I collegamenti ipertestuali che facilitano la navigazione all'interno del documento sono ammessi, ma non sono ammessi collegamenti ipertestuali a parti interne di altri documenti.

12. I collegamenti ipertestuali a siti internet e ai loro contenuti non sono formalmente vietati, ma sono certamente sconsigliabili vista la loro scarsa affidabilità e persistenza nel tempo.

Non si tratta di vincoli da poco e, completato un documento da archiviare, è necessario disporre di strumenti adeguati per verificarne la conformità con le norme.

22.2 Le immagini

Le immagini incluse in un file archiviabile devono essere tutte conformi allo stesso modello di colore. Se si dispone di immagini in vero bianco e nero (senza tonalità di grigio), questo può essere accettato insieme agli altri modelli di colore, ma solitamente le immagini con così poca informazione (ovvero con un contrasto così alto) sono assai scadenti, quindi sarebbe meglio evitarle, specialmente se sono ottenute tramite uno scanner. È sempre possibile configurare lo scanner utilizzato per evitare l'uso di un solo bit per ogni pixel; sarebbe meglio configurarlo per un modello di colore a tonalità di grigio (8 bit per ogni pixel); meglio ancora è lasciare le impostazioni di default che si riferiscono in generale al modello di tricromia additiva RGB (*Red, Green, Blue*) che rendono perfettamente i colori (anche il nero) sugli schermi dove i puntini luminosi emettono luce corrispondente a quei tre colori fondamentali.

Talvolta si è portati a pensare alla stampa, la quale usa la quadricromia sottrattiva CMYK (*Cyan, Magenta, Yellow, blacK*), dove i puntini colorati sono ottenuti depositando particelle piccolissime di colore, che riflettono il colore desiderato assorbendo gli altri colori presenti nella luce bianca con cui viene illuminato lo stampato. È meglio non fare riferimento al modello CMYK; ogni stampante, a seconda del sistema di stampa che usa, del tipo di pigmenti colorati, eccetera, ha il suo convertitore di colore che trasforma il modello additivo RGB in quel che occorre alla stampante per stampare correttamente rendendo i colori al meglio. La stampa è regolata dalla norma PDF/X, ma riguarda solo la stampa, non l'archiviabilità; le norme PDF/A vietano l'uso di profili di colore CMYK o, per lo meno, impongono ulteriori rigide restrizioni.

Per cui tutte le immagini oltre a essere conformi allo stesso modello di colore, è bene che facciano riferimento solo al modello additivo RGB, e non facciano assolutamente uso del modello CMYK.

Questo implica che i *metadati* contenuti nel file PDF archiviabile menzionino esplicitamente il modello di colore. Ne deve specificare non solo il tipo, ma anche il file che lo descrive; questo tipo di file ha estensione `.icc`[1].

Come vedremo successivamente, il pacchetto ***pdfx*** che provvede a tutte le necessità delle varie norme ISO, fornisce anche alcuni file `.icc` per agevolare

[1] File un po' anziani potrebbero avere estensione `.icm`, ma sarebbe meglio evitarli e fare riferimento a file più recenti. L'utente LaTeX non se ne preoccupi, perché i mezzi messi a loro disposizione provvedono direttamente a queste necessità.

l'opera del compositore. Se tuttavia questi desiderasse o volesse usare modelli di colore diversi da quelli preimpostati, potrebbe esaminare il contenuto del proprio elaboratore e vedere se fosse disponibile il profilo di colore che desidera specificare. Se una ricerca sui file del proprio elaboratore non desse nessun risultato utile, allora bisogna che il compositore cerchi in rete quello che desidera; il pacchetto `pdfx` permette poi di impostare le informazioni corrette per il profilo di colore specificato.

Una attenzione particolare va posta ai modelli di colore e ai colori usati per i comandi interni dei programmi di composizione del sistema TEX; per esempio a chi scrive è successo di configurare con il colore *magenta* i link delle citazioni bibliografiche, constatando che il file non era conforme alle norme ISO; cambiando il colore in *blue* il file diventava conforme.

Questo succede perché in mancanza di configurazioni globali, i colori che si possono usare vengono tratti dai comandi interni che fanno uso del minimo di informazione necessaria; il colore *blue* è uno dei colori fondamentali del modello di colore RGB, mentre il *magenta* è uno dei colori fondamentali del modello di colore CMYK, e da questo viene tratta la codifica interna del colore da usare, chiaramente incompatibile con il modello RGB con cui era dichiarato il file e tutte le sue immagini. Basta usare correttamente i comandi di configurazione del pacchetto `color`, o `xcolor`, o `graphicx` e il problema svanisce anche con il colore *magenta* e gli altri colori fondamentali del modello CMYK.

22.3 I font

I file PDF archiviabili devono contenere tutti i font di cui fanno uso; quando se ne guardino le caratteristiche, per esempio, usando il programma gratuito Adobe Reader chiedendogli di mostrare le proprietà del documento, essi devono risultare tutti *Embedded*. In generale i font non vengono incorporati completamente, ma ne viene incorporato un sottoinsieme (*subset*, cosicché l'apposita finestra del Reader indicherà *Embedded subset*) per quei font di cui sono stati incorporati solo i caratteri effettivamente usati nel documento.

Ma attenzione: quando si incorpora una figura, anche PDF, che a sua volta contiene altri font, anche questi devono risultare incorporati. Ci si dimentica assai spesso che le figure prodotte con altri programmi e salvate in formato PDF, spesso non incorporano tutti i caratteri, in particolare i 35 font che si suppone esistano di default su qualunque macchina capace di visualizzare[2] i file PostScript; oppure gli 11 font che si suppone esistano di default su qualunque macchina capace di visualizzare i file PDF.

Bisogna quindi ricordarsi di configurare correttamente i programmi del sistema TEX, affinché incorporino *tutti* i font usati, anche quelli di default[3]; ma

[2] Qui il verbo *visualizzare* è usato anche quando il processo di visualizzazione consiste nella stampa.

[3] Questa è l'impostazione predefinita delle distribuzioni del sistema TEX recenti; non lo era fino a qualche tempo fa; è quindi meglio accertarsene ed eventualmente aggiornare l'installazione

bisogna ricordarsi di configurare anche i programmi di disegno affinché salvino le loro immagini in un formato che contenga almeno il sottoinsieme dei caratteri usati.

I programmi del sistema TEX sono configurati attraverso il file `updmap.cfg`; nelle prime, diciamo, 100 righe ci sono le impostazioni booleane per l'incorporazione dei font, sia per il programma dvips sia per i programmi pdftex e dvipdfm; per questi ultimi due è impostato il valore `true` per l'incorporazione dei font, mentre per il primo potrebbe essere presente l'impostazione `false`; quindi bisogna per prima cosa cambiare questo valore in `true` e poi rilanciare il programma eseguibile updmap (o updmap-sys) al fine di creare le mappe dei font configurate correttamente.

Per le immagini incorporate o si riesce a configurare correttamente il programma di disegno usato, oppure bisogna ricorrere ad un trucco, non altrettanto efficace dal punto di vista estetico, ma accettabile. Il file PDF prodotto esternamente, può venire elaborato con il programma gratuito inkscape chiedendogli di salvare il risultato di nuovo in formato PDF; i font non incorporati, ma resi visibili attraverso i font in dotazione a questo programma, vengono sostituiti con i loro disegni vettoriali; la natura vettoriale dell'immagine presente nel file originale viene conservata, ma dopo il trattamento il file ufficialmente non contiene più nessun font.

Alternativamente, pur di usare una densità di pixel almeno di 300 px/pollice, si può trasformare il file PDF in un file PNG con uno dei programmi di fotoritocco disponibili, per esempio gimp, in modo che l'intera immagine, comprese le sue didascalie, sia trasformata in una matrice di pixel; si perde la vettorialità, ma si tolgono le incompatibilità con le norme ISO. L'utente si ricordi che i file vettoriali trasformati in file bitmapped possono diventare enormemente grandi in termini di numero di byte impegnati per rendere il contenuto del file.

Attenzione: il formato grafico PNG è *lossless* quindi tutta l'informazione dell'immagine è sempre conservata, ma il formato PNG consente anche le "trasparenze"; queste sono vietate nei file da archiviare, quindi bisogna sempre verificare se un dato file PNG contenga delle trasparenze, ed eventualmente bisogna toglierle se il programma di fotoritocco in uso dispone di questa funzionalità. Altrimenti bisogna convertire le immagini in formato JPG che non dispone di trasparenze; però il metodo di compressione del formato JPG è *lossy*, vale a dire che viene persa parte dell'informazione contenuta nell'immagine e, nella ricostruzione dell'immagine per la sua visualizzazione, possono apparire degli artefatti; il problema è virtualmente assente quando l'immagine è una fotografia, ma può essere importante quando si tratta di disegni tecnici o comunque dei disegni "al tratto". Questo fatto consiglia di riservare il formato PNG a questo tipo di immagini, curando che non abbiano trasparenze.

Questo è il primo problema che si presenta con i font; il secondo problema è che ogni carattere deve avere una larghezza non nulla. Sembra ovvio, ma le cose non stanno così.

---

o modificare le impostazioni.

Infatti, specialmente con i font vettoriali (in formato `.pfb`) delle collezioni del sistema TEX, alcuni segni matematici vengono composti giustapponendo elementi che talvolta devono sovrapporsi ad altri elementi. Fra i font che presentano questo *difetto* ci sono tutti i font matematici `cmsy` e i loro parenti di altre collezioni codificati come i font Computer Modern. I segni problematici sono `\not` e `\mapsto`. La presenza anche di una sola istanza di uno di questi segni in un file altrimenti ineccepibile, rende il file PDF non conforme alle norme ISO e non è archiviabile, anche se a schermo e a stampa esso appare perfetto.

Nella descrizione del pacchetto *pdfx*, di cui si parlerà fra non molto, e nel sito `http://support.river-valley.com/wiki/index.php?title=Generating_PDF/A_compliant_PDFs_from_pdftex`, vengono suggeriti alcuni modi di aggiustare la situazione; chiaramente il lettore può seguire quei consigli; qui si presenta una soluzione alternativa, non esattamente equivalente, ma di validità generale e che non richiede la "manomissione" dei file appartenenti al sistema TEX; se e quando i curatori di quelle collezioni di font vorranno provvedere ad eseguire nativamente le correzioni necessarie, non occorrerà più ricorrere a nessun aggiustamento. Invece i font matematici OpenType distribuiti con il sistema TEX non presentano questo difetto, quindi, senza ricorrere a nessun trucco o "manomissione", basta comporre il documento con lualatex evitando, perciò, i font matematici usati da pdflatex che, senza correzioni, sono incompatibili con le norme ISO.

Il comando `\not` viene sovrapposto a qualunque operatore matematico di relazione per negarlo: tipicamente `\not=`, o il suo equivalente `\neq`, produce il segno $\neq$, ottenuto sovrapponendo il segno / al segno =. Ai tempi in cui nacque il sistema TEX le memorie RAM e i clock che regolavano l'esecuzione delle istruzioni macchina, anche nei grossi *main frame*, erano rispettivamente modeste e assai lenti; risparmiare su qualunque cosa era importante, per cui i segni matematici di relazione negati erano tutti ottenuti in questo modo; i font dell'American Mathematical Society hanno rimediato al brutto aspetto di alcuni di questi segni negati ma, anche per compatibilità con i milioni di documenti scritti in passato, il meccanismo è rimasto anche oggi che sono passati più di 30 anni dalla nascita del sistema.

Siccome il segno `\not` privo di dimensioni orizzontali non è accettabile dalle norme ISO, bisogna provvedere a sostituirlo con un trucco di programmazione base di TEX, consistente nel comando primitivo `\mathchoice`; questo comando per ciascuna delle quattro modalità di composizione della matematica (in display, in linea con il testo, indici primi, indici secondi) offre quattro modalità distinte di operazione che vengono specificate ordinatamente nei suoi quattro argomenti. La "correzione" consiste nel comporre dentro una scatola di larghezza nulla, una barra di dimensioni orizzontali non nulle, specificandone una dimensione diversa per ciascuno dei quattro modi. Ecco il codice, dove si sono volutamente incolonnate le corrispondenti graffe aperte in modo da rendersi conto più facilmente delle quattro scelte:

```
\renewcommand*\not{%
\mathrel{%
```

```
\mathchoice%
   {\rlap{$\displaystyle\mkern2.5mu\mathnormal{/}$}}%
   {\rlap{$\textstyle\mkern2.5mu\mathnormal{/}$}}%
   {\rlap{$\scriptstyle\mkern2.5mu\mathnormal{/}$}}%
   {\rlap{$\scriptscriptstyle\mkern2.5mu\mathnormal{/}$}}%
     }%
               }%
```

Il comando `\rlap` (*right overlap*) serve per comporre una scatola di larghezza nulla il cui contenuto si sovrappone a qualunque cosa si trovi alla sua destra; siccome al suo interno il modo di composizione torna ad essere quello testuale, bisogna riattivare il modo matematico e al suo interno, per ciascuna delle quattro modalità di composizione, bisogna specificare lo stile di composizione rispettivamente mediante i comandi `\displaystyle`, `\textstyle`, `\scriptstyle` e `\scriptscriptstyle`.

Il comando `\mapsto` di per sé non sarebbe problematico e produce il segno $\mapsto$. Il problema nasce dalla sua coda "ı" che è definita con una dimensione orizzontale nulla; non è chiaro il motivo, in quanto esso compare solo nella definizione di `\mapsto`; sarebbe stato forse preferibile disegnare il segno direttamente senza ricorrere alla coda di larghezza nulla aggiunta ad una freccia normale; chissà, forse nelle prime versioni del sistema TEX questo segno serviva anche ad altro. Il fatto è che esso non è accettabile in un file conforme allo standard PDF/A.

In questo caso si è preferito usare i comando `\rule` di LATEX; ahimè, questa soluzione è usabile solo con LATEX e ne è escluso l'uso con Plain TEX[4]. Tuttavia... questa Guida si riferisce all'uso di LATEX per comporre tipograficamente in modo eccellente. La soluzione trovata ricorre ancora a `\mathchoice` ma il segno è ottenuto con dimensioni diverse per i quattro modi di composizione distinti:

```
\renewcommand\mapstochar{\mathrel{\mathchoice
{\rlap{\rule[0.05ex]{0.1ex}{1ex}}\mkern-0.5mu}%
{\rlap{\rule[0.05ex]{0.1ex}{1ex}}\mkern-0.5mu}%
{\rlap{\rule[0.035ex]{0.08ex}{0.75ex}}\mkern-0.5mu}%
{\rlap{\rule[0.025ex]{0.065ex}{0.55ex}}\mkern-0.5mu}%
}}
```

I numeri che compaiono nel codice servono per legare lo spessore del segno e la posizione precisa a seconda dello stile di composizione matematico; sono tutti valori ricavati sperimentalmente e, ovviamente, non si tratta di prescrizioni immutabili, ma vanno adattate anche a font diversi da quelli qui ipotizzati dei simboli matematici della collezione Computer Modern.

Queste due soluzioni non toccano i font originali; possono essere inserite in un file di opzioni personali da usare sempre, o soltanto quando si devono produrre file PDF archiviabili. Speriamo solo che in un lasso di tempo ragionevole queste correzioni, o meglio correzioni più professionali vengano apportate ai font

[4] O meglio, il comando `\rule` di LATEX deve venire sostituito dal comando `\vrule` di TEX corredandolo delle necessarie specificazioni di altezza, larghezza e profondità.

originali e vengano ridefiniti i comandi di generazione di questi segni, in modo che il risultato finale sia lo stesso, ma sia anche compatibile con il formato PDF archiviabile.

Vale la pena segnalare la disponibilità nel sistema TEX del font Libertinus assieme al corrispondente file di font matematici dove questo difetto di `\not` e di `\mapstocar` è assente.

Al contrario, potrebbe succedere che altri file che fanno uso di altri font possano dare luogo a problemi simili; a seconda della diagnosi eseguita con Acrobat Professional o con VeraPDF, ci si comporterà adeguatamente, ma in modo analogo a quanto descritto sopra.

22.4 Gli hyperlink

Anche gli hyperlink possono dare dei problemi, ma il pacchetto *hyperref* accetta l'opzione *pdfa* che consente di evitare questi problemi. L'unico che resterebbe è quello di ricordarsi di specificare l'opzione, ma a questo scopo il pacchetto *pdfx* provvede da solo; se per qualche motivo non fosse possibile usare pdflatex (o lualatex), non si potrebbe usare *pdfx* e bisognerebbe usare procedimenti indiretti per produrre il file PDF/A desiderato, per esempio attraverso il pacchetto *pdfpages* o il programma ghostscript; in questo caso bisogna ricordarsi di specificare bene l'opzione alla chiamata esplicita di *hyperref*.

Vale la pena di ricordare che nel formato PDF/A gli hyperlink interni continuano ad essere attivi, ma gli hyperlink verso indirizzi esterni, pur restando colorati come qualunque altro hyperlink, sono disattivati (almeno i programmi della Adobe li disattivano se hanno il sospetto di star gestendo un file PDF/A); la ratio di questa limitazione è che già oggi si incontrano molti hyperlink che non funzionano più; probabilmente fra cinquanta anni nessuno degli indirizzi internet oggi esistenti sarà più valido. Sarebbe quindi perfettamente inutile indirizzare i lettori verso indirizzi 'morti'. Bisogna tenere conto di questo fatto, specialmente quando si scrivono i riferimenti nella bibliografia o nel testo, in modo da evitare riferimenti ad indirizzi internet esterni; per altro nella bibliografia, secondo norme recenti, è diventato obbligatorio indicare l'ultima data di consultazione del sito in modo che il lettore sia avvisato che in quella data il link era corretto, mentre nella data della sua lettura quel link potrebbe non esserlo più. Oggi esiste la WayBack Machine dell'Internet Archive, dalla quale si potrebbe, forse, recuperare quella pagina, anche se il link diretto non funziona più.

22.5 Generazione di un file PDF archiviabile

Affrontati i problemi a cui bisogna porre attenzione (che riflettono un po' anche la strada tortuosa che chi scrive ha dovuto scoprire "sperimentalmente", visto che la documentazione era e resta (per ora) spiacevolmente carente) non resta che procedere alla generazione di un documento in formato PDF archiviabile.

Negli anni la documentazione del pacchetto *pdfx* è migliorata molto, quindi oggi le difficoltà sono diminuite. Ma se si deve ottenere un file conforme alle norme PDF/A con altri programmi, compreso ghostscript, l'insuccesso è quasi garantito. Recentissimamente (2024) i programmi di composizione sono stati dotati di un comando `\DocumentMetaData` che permette di ottenere molto facilmente un file PDF/A-1b senza bisogno di ricorrere ad altri pacchetti.

22.5.1 La strada maestra

La strada maestra valida ancora oggi, anche se oggi, come detto sopra, quei compilatori possono funzionare da soli purché sia stato usato il comando `\DocumentMetaData`, è quella di usare pdflatex o lualatex richiamando il pacchetto *pdfx*. Questo pacchetto è parte integrante di ogni distribuzione completa del sistema TEX; si suggerisce la lettura attenta della sua documentazione, tenendo conto anche di ciò che è detto qui.

L'unica cosa che non è scritta nella documentazione di pdfx è che questo pacchetto deve essere caricato nel preambolo come prima cosa[5] e *prima* di *hyperref*, che per altro va caricato senza opzioni; *hyperrref* può venire successivamente configurato mediante il suo comando `\hypersetup`; in realtà *hyperref* viene già caricato da *pdfx*, ma siccome questo pacchetto potrebbe venire caricato in un secondo momento, quando la lavorazione del documento è quasi terminata, non è vietato caricarlo prima della preesistente chiamata a *hyperref* purché questa sia eseguita senza opzioni.

Il pacchetto, solo in congiunzione con con pdflatex, o con lualatex, serve per produrre file in formato PDF/A-1b se gli specifica l'opzione *a-1b* oppure file in formato PDF/X-1a se gli specifica l'opzione *x-1a*; il formato PDF/A-1b è quello che ci interessa qui. Il formato PDF/X-1a serve per rispettare un'altra norma ISO indirizzata alla conformità dei file stampabili; le problematiche sono simili, ma i file stampabili sono completamente orientati al modello di colore CMYK e dispongono di una serie differente di *metadati*. Il pacchetto *pdfx* in realtà serve per produrre file conformi con un'altra dozzina di norme derivate dalla ISO 19005, ma qui ci soffermiamo sulla PDF/A-1b perché è quella che ci interessa in questa guida. Le norme più restrittive che contengono la lettera 'a' minuscola, come per esempio PDF/A-1a, potrebbero essere ottenibili se si disponesse nel sistema TEX di un programma che produca file PDF di tipo "tagged PDF"; questo non è ancora sempre fattibile, ma un apposito gruppo di lavoro sta lavorando su questo problema; in realtà sembrerebbe che chi lavora sul mark up ConTEXt abbia già ottenuto risultati in questo campo; qui non ne parliamo perché questa guida si riferisce a quanto si può fare con il mark up LATEX. Il pacchetto sperimentale

[5] Se l'utente vuole servirsi del pacchetto *imakeidx* per comporre uno o più indici analitici in modo sincrono con la compilazione del documento, bisogna ricordare che questo pacchetto va caricato *prima* di *hyperref*; siccome questo è caricato da *pdfx*, ne segue che *imakeidx* deve essere caricato prima di *pdfx*. Dimenticarsi di questo fatto non impedisce di generare e compilare i vari indici analitici, ma i numeri di pagina indicati a fianco di ogni lemma non risultano collegati in modo ipertestuale alle pagine a cui l'indice rinvia. Poco male per un testo da stampare, ma molto scomodo per un testo da leggere a schermo.

tagpdf è già distribuito come versione beta nelle distribuzioni più recenti del sistema TEX; si rinvia il lettore alla sua documentazione leggibile con `texdoc tagpdf`; ma qui non se ne dice di più, per il semplice motivo che chi scrive non ne ha ancora (2024) esperienza sufficiente.

Il punto essenziale, che invece non bisogna dimenticare, serve per impostare i *metadati* relativi al documento da archiviare. Questo va fatto evidentemente per ogni documento da archiviare e sempre con dati diversi. La cosa migliore da fare è quella di scrivere ogni volta un file con estensione `.xmpdata` e con il nome coincidente con il file principale (o unico) che contiene il testo sorgente `.tex` da comporre. Nel caso di questa Guida, per esempio, il cui file principale si chiama `GuidaGuIT.tex`, bisogna predisporre un file dal nome `GuidaGuIT.xmpdata` e con il contenuto corrispondente alla sintassi seguente:

```
\Title{⟨Titolo del documento⟩}
\Author{⟨Lista degli autori⟩}
\Keywords{⟨lista delle parole chiave⟩}
\Org{⟨Organizzazione che provvede alla pubblicazione⟩}
```

Si possono inserire anche altri comandi (tutti inizianti con una lettera maiuscola) e se ne può vedere l'elenco completo nella documentazione del pacchetto *pdfx*. Una limitazione dei metadati è che dovrebbero contenere solo caratteri ASCII; le recenti versioni del pacchetto consentono anche di usare qualunque codifica d'entrata a 8 bit, ma si incontrerebbero problemi non indifferenti, perché non sono "universali"; invece la codifica UTF-8 sarebbe quella da preferire se quella ASCII non à adeguata per scrivere le stringhe alfabetiche dei vari argomenti.

Non c'è molto da dire sul significato degli argomenti; si noti che la lista delle parole chiave sarebbe meglio che fosse una lista strutturata, non semplicemente una lista di parole separate da virgole; tenuto conto che l'apertura e la chiusura della struttura sono già inserite di default, quello che serve è solo mettere il delimitatore `\sep` fra una parola chiave e l'altra; la sintassi da usare è perciò è la seguente:

```
⟨prima chiave⟩\sep
⟨seconda chiave⟩\sep
...
⟨penultima chiave⟩\sep
⟨ultima chiave⟩
```

Il modo suggerito dalla documentazione di *pdfx* per creare il file di metadati con il nome consistente e l'estensione corretta, è quello di includere i soli metadati nell'argomento dell'ambiente *filecontents* inserito prima[6] dell'istruzione `\documentclass`, come mostrato nella pagina 574.

[6] Una recente modifica del nucleo di LATEX ridefinisce l'ambiente *filecontents* anche nel preambolo; questo nuovo ambiente accetta anche diverse opzioni molto utili; si legga la documentazione di *filecontents*, che descrive come usare il nuovo ambiente, anche se il pacchetto *filecontents* non esiste più in quanto le sue funzionalità sono già contenute nel nucleo di LATEX.

```
\begin{filecontents}[noheader,overwrite]{\jobname.xpmdata}
\Title{⟨Titolo del documento⟩}
\Author{⟨Lista degli autori⟩}
\Org{⟨Organizzazione che provvede alla pubblicazione⟩}
\Keywords{%
⟨prima chiave⟩\sep
⟨seconda chiave⟩\sep
...
⟨penultima chiave⟩\sep
⟨ultima chiave⟩}
\end{filecontents}
\documentclass[⟨opzioni⟩]{⟨classe⟩}
```

Compilando il documento, alla variabile `\jobname` viene automaticamente assegnato come valore il nome del file sorgente, per cui in questo modo si è sicuri che estraendo il contenuto dell'ambiente *filecontents* questo venga salvato in un file con il nome giusto e nella stessa cartella dove si trova il file in compilazione.

Bisogna eseguire pdflatex (o lualatex; ma vedi dopo) quante volte occorre per soddisfare anche le esigenze della bibliografia e dell'indice analitico e poi bisogna sottoporre a verifica di conformità il file ottenuto. Scaramanticamente sarebbe opportuno incrociare le dita. Non è facile sapere se si è riusciti ad evitare tutti i trabocchetti con i *metadati*, le immagini, i colori e i font che potrebbero produrre un file non conforme; la lettura della diagnosi eseguita dal programma di verifica consente di focalizzare l'attenzione su qualche problema residuo. Tuttavia chi scrive ha già prodotto molti file archiviabili e ormai ha sviluppato una piccola esperienza che gli permette di affermare che se si seguono le procedure descritte sopra, le possibilità di incontrare problemi di conformità sono ridotte; in ogni caso si è indicata la strada per affrontare alcuni problemi con i font e con i file inclusi.

22.5.2 Trasformazione mediante il file *pdfpages*

Se si è disposti a rinunciare anche ai collegamenti ipertestuali interni, si può trasformare un file PDF prodotto con xelatex in un file PDF/A. Siccome il motore di composizione xetex non dispone delle funzionalità specifiche di pdftex, esso non può produrre direttamente un file PDF/A, ma può produrre un file PDF che contenga solo font OpenType, perfettamente accettabili dal formato PDF/A, e figure col profilo di colore RGB; naturalmente bisogna prestare attenzione a quelle immagini che contengono legende perché devono contenere i font usati, oppure devono venire convertite come si è spiegato sopra affinché non usino nessun font ma siano immagini pure.

Ottenuto il file PDF, supponiamo con il nome `XeMain.pdf`, se ne predispone un altro con un nome qualsiasi, supponiamo `pdfMain.tex` che accompagneremo

con il file di *metadati* `pdfMain.xmpdata` contenente i dati richiesti, come si è indicato nel paragrafo 22.5.1. Il nuovo file sarà così composto:

```
% File pdfMain.tex
\documentclass{book} % o la stessa classe di XeMain.tex
\usepackage[a-1b]{pdfx}
\usepackage{pdfpages}
\begin{document}
\pagestyle{empty}
\pdfinclude[pages=-]{XeMain.pdf}
\end{document}
```

Lo si compilerà con pdflatex, ottenendo il file `pdfMain.pdf`. Se non si presentano quei trabocchetti a cui si è fatto cenno, il file dovrebbe risultare conforme alla norma PDF/A-1b. La verifica della conformità è comunque necessaria, ma dovrebbe dare un responso positivo.

22.5.3 Trasformazione di un file PDF o di un file PS

Il programma ghostscript permette di convertire un file PS in un file PDF e l'uno e l'altro in file conformi agli standard PDF/A-1b oppure PDF/X-1a. Attenzione: questa possibilità di conversione si ha dalla versione 8.60 di ghostscript in poi.

Tuttavia i risultati che si possono ottenere seguendo questa via non sono molto diversi da quelli che si possono ottenere con il metodo descritto nel paragrafo 22.5.2; caso mai si può incontrare qualche problema in più. Per questo motivo qui non si descrivono le operazioni da eseguire per usare il metodo di ghostscript ma si rinvia il lettore alla sua stessa documentazione. Uomo avvisato, mezzo salvato, dice il proverbio: infatti quella documentazione è abbastanza nebulosa e richiede da parte dell'utente doti di intuizione non indifferenti. Chi scrive, pur avendo usato questo metodo per anni, ritiene che ora il gioco non valga più la candela.

22.5.4 La produzione mediante lualatex

Nella documentazione di *pdfx* c'era scritto (alla fine del 2015) che il pacchetto non poteva essere usato con lualatex. Era vero nel 2015, ma nel 2016 è uscita la nuova versione di *pdfx* che consente di produrre file PDF/A compatibili direttamente anche con lualatex, senza ricorrere ai trucchi che erano necessari nel 2015.

In sostanza bisogna procedere con i metadati come si è descritto nel paragrafo 22.5.1; cioè inserire i metadati nell'ambiente *filecontents* con il nome del loro file specificato tramite la variabile `\jobname` e poi caricare il pacchetto *pdfx* come prima cosa nel preambolo. Se si vuole caricare anche il pacchetto *hyperref*, lo si può fare, però senza specificargli nessuna opzione, ma eventualmente specificandogliene in un secondo tempo mediante l'argomento del comando `\hypersetup`.

Di problemi con i font non ce ne sono se si usano solo font OpenType. I problemi con le immagini incorporate sono gli stessi che si devono affrontare e superare con pdflatex.

Bisogna usare una piccola avvertenza: il file `.xmpdata` con i metadati che bisogna affiancare al main file del documento da comporre, non può accettare altro che caratteri ASCII o UTF-8, e il documento deve avere tutti i suoi file principale e secondari codificati in UTF-8. Non dovrebbero quindi esserci problemi di sorta, ma bisogna ricordarsi che il file `.xmpdata` deve essere salvato anch'esso o con soli caratteri ASCII o con caratteri conformi alla transcodifica UTF-8.

Chi scrive ha lavorato molto con i file da archiviare ed ha tirato un sospiro di sollievo quando il pacchetto *pdfx* è diventato compatibile anche con tutti i programmi del sistema TEX; tenuto conto che un file predisposto per funzionare con pdflatex è compilabile anche con lualatex dopo le modifiche descritte in alcuni capitoli precedenti (riguardanti la gestione delle lingue e la gestione dei font), ora esso ritiene che oggi la via migliore per produrre un file archiviabile sia proprio quella di comporlo con lualatex, perché i problemi con i font usati nel documento spariscono completamente (restano però quelli dei font contenuti nei file delle immagini importate). In sostanza non c'è quasi nulla da fare in più rispetto a quanto sarebbe necessario fare usando pdflatex, quindi usare lualatex, a conti fatti, risulta vantaggioso.

22.6 Verifica della conformità

Chi scrive ha provato ad ottenere la verifica della conformità ricorrendo a servizi (gratuiti) in rete, ma non ha mai ottenuto diagnosi positive con i tipi di documenti che ha inviato per la verifica. Egli invece ha ottenuto ottimi risultati con le verifiche eseguite con Acrobat Pro XI[7], con Acrobat 2020 e con VeraPDF, i primi due programmi commerciali, il secondo programma gratuito.

Il programma progAcrobat dispone di un modulo che si chiama Preflight che consente di eseguire la verifica e di produrre la certificazione di conformità dei file che dovrebbero rispettare le numerose norme PDF/A per i documenti archiviabili e PDF/X per i documenti stampabili. La documentazione di conformità è molto dettagliata e riguarda ogni possibile aspetto. Se la diagnosi fosse negativa, la lettura di questa documentazione permette di andare rapidamente al punto dolente per provvedere alla sua cura.

Il terzo programma è molto più semplice da usare (volendo, anche in rete) anche perché serve solo per la diagnosi e verifica dei file. Per l'installazione sul proprio elaboratore richiede una certa esperienza con l'uso dei calcolatori per installarlo con tutte le librerie di cui ha bisogno. Tuttavia, una volta installato, con quattro click di mouse si carica il file da esaminare e certificare, lo si esamina e si stampa (o si salva in un file) il rapporto della diagnosi, meno dettagliata di quella prodotta da Preflight, ma comunque con tutte le informazioni necessarie.

[7]Non più acquistabile da diversi anni.

Si suggerisce di riferirsi per i dettagli alla guida tematica *Creare file archiviabili con pdflatex e lualatex*, nella sezione delle Guide tematiche interna alla sezione Documentazione accessibile dal Forum del GUIT. Tra l'altro in questa guida si parla più diffusamente del procedimento usabile con `\DocumentMetaData`; interessante e utile.

22.7 Commenti

Il problema è difficile ma oggi (2024) con i mezzi a disposizione degli utenti di TEX Live le cose sono diventate più semplici.

Si tenga presente che gli stessi problemi si pongono con la suite Microsoft Office 2007 e con OpenOffice.org e LibreOffice nella versione 3.x; persino il programma principe della Adobe, Acrobat Pro versione 8.x, incontrava i suoi problemi. Oggi Acrobat 2020 non sembra avere più i problemi di qualche anno fa, ma questa vicenda la dice molto lunga sulla difficoltà di creare file PDF/A-compatibili. Ancora nel 2016 veniva segnalato che presso il French National Archive, per le tesi di dottorato (prodotte in PDF con i programmi del sistema TEX), esistevano dei problemi di verifica della conformità anche se per Acrobat Pro risultavano conformi. Siccome si trattava di software proprietari è difficile dire se era solo di una questione di diversa tolleranza verso piccoli errori, oppure se c'erano dei "bachi" in questo o quel software.

A chi scrive (e che si è già fatto una certa esperienza in merito) appare più corretto non nascondere le difficoltà e i problemi, che non proporre agli utenti programmi, spesso piuttosto costosi, che non possono fare più di tanto e che mancano il bersaglio più spesso di quanto non si creda.

Dispiace che il bersaglio non sia sempre centrato nemmeno con OpenOffice.org o con LibreOffice, che, se non altro, hanno due pregi: (*a*) fin dai loro inizi hanno usato come formato di default quello con il mark-up XML, e (*b*) si tratta di programmi gratuiti e aperti in mano alla comunità degli utenti, non molto diversamente da quel che succede con il sistema TEX, pur restando su un terreno ben diverso[8]. È quindi un peccato che anche OpenOffice.org e LibreOffice siano scivolati su questa questione, senza una parola di avviso relativa alla difficoltà di salvare file veramente conformi alla norma ISO.

Merita segnalare che tanto Microsoft Office quanto OpenOffice.org e LibreOffice consentirebbero di esportare i loro documenti in formato PDF; viene chiesto all'utente se desiderano la versione "semplice" o se desiderano la versione archiviabile vuoi secondo la norma "1b" vuoi secondo la "1a". A chi scrive, che ha fatto numerosi esperimenti con tutte e tre queste collezioni di programmi per la

[8]La matematica è sempre stato un problema difficile da affrontare con programmi di tipo WYSIWYG; OpenOffice.org ha però un plug-in che consente di comporre le formule con LATEX; è sulla buona strada ma, se uno deve avere sulla propria macchina anche il sistema TEX per comporre le formule di OpenOffice.org, a che cosa serve usare OpenOffice.org? Solo per poter usare la tecnica WYSIWYG? Non so se la notizia sia vera, ma sembra che il nostro Grand Wizard, Donald E. Knuth, abbia collaborato con la Microsoft per l'implementazione dell'*equation editor* della versione 2007 di MS Office

produttività individuale, non è mai successo che un file PDF, creato con quelle suite ed esportato secondo una delle due specifiche PDF/A, passasse la verifica di Preflight; può darsi che con alcuni documenti puramente testuali ci si riesca, ma a lui non è mai capitato.

Va menzionato un altro punto. I programmi della Adobe, il Reader XI e l'Acrobat Pro XI, riconoscono se un file PDF contiene i metadati per poter essere conforme alle norme PDF/A. La presenza dei metadati però è necessaria ma non sufficiente per garantire che il file sia conforme a quelle norme. Ciò nonostante entrambi i programmi segnalano che il file appena aperto potrebbe essere conforme con le norme PDF/A, perciò il file viene aperto in modalità "read only". È vero; è prudente avvisare l'utente del fatto che il file *potrebbe* essere conforme e una sua possibile modifica, anche le piccole modifiche che si potrebbero fare con il Reader, potrebbe compromettere la sua eventuale conformità con le norme. Ma è anche vero che quell'affermazione, ben evidenziata in una barra azzurra sotto le barre principali dei due programmi, indurrebbe a pensare che il file sia davvero conforme. Solo il modulo Preflight di Acrobat Pro XI 0 2020 o il programma VeraPDF possono dire con certezza se il file sia conforme per davvero. Non è quella barra azzurra con quel che c'è scritto dentro che certifica la conformità con le norme. Bisogna quindi fare molta attenzione a non lasciarsi fuorviare. La cosa è importante sia per le persone che devono depositare documenti ufficiali conformi, sia per coloro, persone fisiche o giuridiche, che devono ricevere tali documenti.

Dal 2019 la versione di Adobe Acrobat Pro XI regolarmente acquistata non viene più aggiornata e sul MacOS X aggiornato non funziona più. Questo perché la Adobe ha cambiato politica e ora il programma Adobe Acrobat Pro DC può essere usato solo in rete dietro il pagamento di un abbonamento mensile o annuale abbastanza oneroso. L'utente che ne facesse un uso saltuario non otterrebbe nessun vantaggio da questo nuovo rapporto economico. È un vero peccato; chissà se in futuro la politica commerciale dell'Adobe tornerà ad essere accessibile anche ai piccoli utenti. L'ultima versione gratuita di Adobe Reader DC continua a fare alcune delle cose che la precedente versione faceva, ma contiene accattivanti inviti ad abbonarsi al servizio on line di Adobe Acrobat Pro DC per le funzionalità già presenti nella precedente versione. Però oggi esiste la versione Acrobat 2020, acquistabile collettivamente a prezzo accessibile da parte di enti istituzionali per i propri dipendenti, sostituisce molto bene la versione economicamente onerosa Acrobat Pro DC; quindi, almeno i dipendenti di quegli enti possono disporre delle enormi funzionalità di questo programma.

Capitolo 23

Comporre documenti di molti autori

Un problema che sorge spesso consiste nel comporre un'opera collettiva dove i vari autori scrivono usando programmi diversi; del tutto simile è la situazione di colui che si trova nella difficile situazione di dover comporre un testo scritto da un altro autore che ha usato uno strumento "scrittorio" diverso da LaTeX.

Occorre convertire il formato dei testi sorgente per introdurli nell'unico master file del documento LaTeX o, più frequentemente, per creare nuovi file da far leggere al master file.

L'evento più frequente è quello che richiede di aggiungere il mark-up di LaTeX in un file relativo ad un documento scritto inizialmente con un word processor.

Naturalmente potrebbe anche succedere di dover fare conversioni opposte, da LaTeX ad un altro formato. Qui non si tratteranno queste conversioni; ci si limita a citare il programma TeX4ht, generalmente facente parte di ogni distribuzione del sistema TeX non minimale o di base. Questo programma può convertire nel formato HTML un documento scritto con il mark-up della versione standard di LaTeX, cioè che non faccia uso di pacchetti di estensione generici, in particolare che non faccia uso di macro personali del compositore. Chi l'ha usato ne dice un gran bene, ma sottolinea la limitazione fondamentale che il file da convertire deve essere strettamente conforme al linguaggio LaTeX di base, qual è descritto nella guida di Lamport [39].

Un programma molto versatile sembra essere Pandoc, [19]. Questo programma, da usare dal terminale, permette di convertire ogni documento scritto virtualmente in qualunque formato in un altro documento scritto, sempre virtualmente, in qualunque altro formato. In realtà i formati in entrata e i formati in uscita non sono tutti i possibili formati, ma le due liste, benché diverse, sono talmente ricche che premettere 'virtualmente' alla parola 'qualunque' non sembra fuori luogo. Mediante opportune opzioni è possibile configurare l'esecuzione del programma per eseguire la conversione in modalità diverse. Fra le altre possi-

bilità esiste la possibilità di convertire un file LaTeX in un file XML, HTML e anche EPUB (sia pure con certe limitazioni). Si rinvia il lettore alla pagina web `http://johnmacfarlane.net/pandoc/` da dove è possibile scaricare sia il programma sia il manuale d'uso.

D'altra parte questa Introduzione tratta della composizione tipografica con LaTeX, quindi appare del tutto ragionevole esaminare come si possa creare un documento compilabile con pdflatex o lualatex o xelatex partendo da una varietà di documenti di varia origine.

23.1 Conversione manuale

La conversione manuale si può dire che richieda non poca fatica, ma è applicabile qualunque sia l'origine dei documenti da comporre e da assemblare assieme in una collettanea. Di solito si tratta di documenti scritti facendo uso di un word processor come Microsoft Word, StarOffice Writer o il Writer della collezione OpenOffice.org (fratello di StarOffice, ma con le caratteristiche del software Open Source) o del fork LibreOffice; potrebbero essere documenti in formato PDF oppure documenti in rete in formato HTML; potrebbero essere scansioni di documenti stampati; potrebbero essere persino dei manoscritti, nel senso letterale del termine: documenti scritti a mano! Insomma, il formato delle informazioni da elaborare può essere veramente eterogeneo.

Mentre un documento Word o Writer potrebbe venire convertito facendo uso di opportuni programmi, gli altri formati potrebbero essere difficili da elaborare in modo automatico; esistono diversi strumenti che consentono di eseguire alcune operazioni preliminari, che facilitano non poco l'elaborazione manuale.

Per i documenti HTML o si toglie il mark-up specifico di quel formato, oppure bisogna convertire il file HTML in un altro file con il formato XML o anche `.docx` e poi procedere con i programmi che verranno esaminati nei prossimi paragrafi.

I documenti scanditi[1] possono essere elaborati attraverso un apparato e un programma di Optical Character Recognition (OCR); sostanzialmente si scandisce di nuovo una stampa del file grafico, frutto della prima scansione, ordinando allo scanner di eseguire il riconoscimento dei caratteri; il risultato può essere molto buono o decisamente deludente; dipende da quanto era stata accurata la prima scansione e dipende molto dal tipo di caratteri usati; il riconoscimento

[1] Questo libro non è certo un modello di purezza linguistica; tuttavia il curatore si rifiuta di usare *scannerizzare* oppure *scansionare*, benché siano i termini più usati. Il verbo *scandire* e il sostantivo che ne deriva, *scansione*, rendono in italiano esattamente la stessa idea. Il nome *scansione* era già usato 50-60 anni fa per descrivere il movimento del pennello elettronico nei cinescopi per eccitare sequenzialmente tutte la particelle fotoluminescenti che avrebbero prodotto l'immagine televisiva. È esattamente il movimento del raggio luminoso che illumina sequenzialmente l'immagine da scandire nell'apparecchio che ha il nome tecnico inglese di *scanner*, ma per il quale il sostantivo italiano *scanditore* o *scansore* farebbe ridere tutti. D'altra parte il latino *scandĕre* vuol dire 'scandire'; *scansione(m)* vuol dire 'scansione'; *scansore(m)* chi o ciò che esegue la scansione. Perché *scansore*, nonostante le sue nobili e antiche origini, dovrebbe far ridere?

automatico riesce meglio con certi caratteri piuttosto che con altri. In ogni caso il file in formato `.txt` (da preferire, anche se lo scanner può produrre anche un file in formato `.docx`) contiene molti errori di ortografia, dovuti al mancato riconoscimento di certi caratteri, e quindi richiede di essere rivisto e corretto con attenzione.

Le immagini presenti nella scansione originale possono solitamente venire estratte, sperando di non degradare troppo la loro qualità, altrimenti devono venire eseguite *ex novo* tutte le immagini costituite da disegni al tratto; mentre per le fotografie o si può accedere agli originali, oppure bisogna accontentarsi. Questo è un problema comune anche quando si deve trasformare un file originario in formato `.docx`.

Comunque, ultimati questi preparativi possiamo supporre di avere a disposizione il testo da trasformare in formato di testo semplice oppure di testo formattato, contenente eventualmente immagini, tabelle, formule, cambiamenti di stile dei font, cambiamenti di margini, e simili; il testo, insomma potrebbe essere in formato `.txt`, oppure `.docx` o `.rtf`, oppure `.pdf`; sì, anche in formato PDF; ogni visualizzatore di questo formato, anche il semplice Adobe Reader, permette di selezionare il testo e di copiarlo per incollarlo altrove; bisogna semplicemente scegliere dalla barra superiore lo "strumento" per selezionare il testo, generalmente caratterizzato da una icona a forma di **T**.

Qualunque metodo si usi per trasformare il file, bisogna ricordarsi che un documento puramente testuale non contiene nessuna informazione descrittiva del testo; un file formattato, oltre al testo, contiene informazioni solamente riguardanti il modo di apparire, non quello che esso rappresenta veramente; il suo modesto mark-up contiene solo poche informazioni sulle immagini, le tabelle, le formule, i font, ma raramente contiene informazioni sul tipo di contenuto di un brano di testo, diversamente da come fa LaTeX. Questo significa che queste informazioni vanno aggiunte a mano dalla persona che si occupa della trasformazione, anche se per alcune parti di questa operazione si avvale di strumenti automatici; questi faranno il possibile, ma certamente non potranno aggiungere informazioni che dipendono dalla comprensione del testo se non sono comprese nel mark-up interno del documento. Bisogna ricordarsene anche quando si discuterà della conversione "semiautomatica".

Ciò premesso, e ben consci del lavoro necessario per una buona conversione, vediamo come eseguire la conversione completamente manuale.

23.1.1 Copia e incolla

Sembra semplice dire che si esegue la trasformazione manuale con la tecnica del copia e incolla; concettualmente non c'è nient'altro da fare, ma ci sono alcune operazioni che conviene tenere presenti.

1. Bisogna predisporre o un master document, o, comunque, una intelaiatura del file LaTeX nel quale avverrà la conversione; dalla dichiarazione della classe alla specificazione dei pacchetti che si intende o si prevede di dover

usare, fino all'intelaiatura dell'ambiente *document*, da riempire via via con il testo da comporre. Se occorressero altri pacchetti, li si può sempre invocare in un secondo tempo, ma è meglio aver studiato prima il documento da convertire per farsi un piano di lavoro destinato agli interventi da eseguire.

2. Conviene procedere un capoverso alla volta; lo si copia dalla finestra dell'applicativo che consente di visualizzare il formato del documento da convertire, e lo si incolla nella finestra di input del file `.tex`. Se c'è qualche intervento da fare, lo si esegue subito: per esempio per mettere in enfasi alcune parole, introdurre le virgolette giuste o i caporali giusti, eventualmente cambiando le virgolette copiate (vedi più avanti i cambiamenti necessari per i caratteri non ASCII); eseguire l'impostazione delle formulette matematiche in linea, e simili piccoli interventi che, proprio perché piccoli, sfuggirebbero se si copiasse l'intero documento in un colpo solo.

3. Per le figure bisogna estrarle dal documento, se non si dispone di file appositi; nel caso della composizione di un testo "nuovo", tutta questa operazione potrebbe essere dovuta all'autore che non conosce LaTeX e che consegna al compositore il testo scritto, per esempio, con Word e, pur avendo inserito le figure nel file `.docx` ha consegnato al compositore anche le foto e le altre illustrazioni mediante file appositamente allegati. Se non fosse così il compositore risparmierà molto tempo se richiede questo materiale all'autore, ricordando che per il materiale a matrici di pixel, come le fotografie, la densità ottimale di punti al pollice deve essere almeno di 300 dpi ma, accettando un lieve degrado, si può accettare anche una densità di 150 dpi; dimezzando la densità, l'ingombro dell'immagine in memoria si riduce di circa quattro volte.

   Se invece non si dispone delle illustrazioni originali, estrarle dal documento vuol dire perdere enormemente in dettaglio e si è fortunati se si riesce ad estrarre immagini a 100 dpi; la qualità scende vistosamente approssimativamente in proporzione al quadrato del rapporto delle densità; scendere da 300 dpi a 100 dpi vuol dire perdere in qualità di un fattore di quasi 10 volte!

4. Per le tabelle si può procedere sia ricomponendo ogni tabella cella per cella, reimpostandola in modo più professionale; oppure la si può copiare integralmente e, dopo averla incollata, se il proprio *shell editor* lo consente, si possono eseguire le sostituzioni di tutti i caratteri `<TAB>` (ASCII 09) con il carattere `&` ricorrendo alle *regular expressions*, un modo avanzato di usare gli *shell editor* che consente di fare delle correzioni selettive anche molto elaborate. Bisogna poi marcare i fine riga della tabella con `\\` e, eventualmente, inserire qualche filetto.

5. Per le formule nel testo si è già detto; per quelle in display o le si ricompone a mano partendo da zero, oppure può succedere che l'*equation editor* usato per comporre il file da trasformare abbia salvato oltre all'immagine della

formula, anche il codice usato per comporla; allora conviene copiare questo codice che spesso e volentieri è codice TEX, o gli assomiglia molto. La collezione di programmi da ufficio OpenOffice.org dispone di un *plug in* che mette a disposizione dell'utente un *equation editor* basato su LATEX; il codice LATEX, quindi, può essere copiato e incollato direttamente. Se si ha la competenza per farlo, si possono anche correggere gli errori delle formule da trasferire sul documento LATEX, sempre prendendone nota e informando l'autore per chiedere conferma della correttezza della correzione. Se non c'è autore a cui chiedere, al massimo si può ripetere la formula originale senza correzione, inserendo la formula corretta in nota accompagnata da un commento adeguato.

6. Le note vanno copiate e incollate durante la lavorazione dei singoli capoversi; è molto più difficile inserire le note alla fine dopo aver sistemato l'intero testo; non è complicato, perché basta incollare il testo della nota fra le due graffe che delimitano l'argomento del comando `\footnote`.

   È molto più delicato convertire le note di tipo umanistico per i riferimenti bibliografici; come è noto gli umanisti sono soliti annotare i riferimenti bibliografici in calce alla pagina usando il meccanismo delle note; questo meccanismo, per conservare spazio, usa spesso abbreviazioni di tipo *Ibidem* oppure *loc. cit.* per riferirsi all'opera citata immediatamente prima oppure non sequenzialmente ma nella stessa pagina. Queste note vanno trattate con comandi `\cite` ben congegnati e in accordo con gli appositi stili bibliografici per la bibliografia. Di conseguenza questo genere di note va convertito avendo bene chiaro in mente il tipo di stile bibliografico e il tipo di citazione, in modo da usare il pacchetto adatto ad affrontare questo tipo di composizione.

7. Per i caratteri non ASCII si può procedere capoverso per capoverso, oppure alla fine dell'introduzione dell'intero testo da convertire; qui le esigenze sono opposte a quelle suggerite sopra per procedere un capoverso alla volta. Infatti i caratteri da convertire sono sorprendentemente numerosi, specialmente quando si converte un file originariamente scritto in formato `.docx`.

   Naturalmente è possibile affrontare il problema con un opportuno encoding dei caratteri in input e un altrettanto opportuno encoding del font con cui comporre. Probabilmente questo è un caso in cui usando sia per il file sorgente sia per il font da usare in uscita l'encoding `utf-8`[2] si risparmia molto lavoro. Chi scrive ha già convertito in passato un numero non indifferente di documenti lavorando "a mano", ma a quei tempi non ha mai usato la codifica `utf-8` non ancora disponibile e meno che mai preimpostata

[2] La codifica in input per usare l'encoding `utf-8` è ottenuta specificando l'opzione *utf8*; per l'uscita sarebbe preferibile usare font codificati in UNICODE; per questo è molto meglio usare il programma xelatex, oppure lualatex, che gestiscono direttamente i font contenenti più di 256 caratteri; le versioni come xelatex e lualatex, che contengono la parte LA, sono generalmente compatibili con il mark-up di LATEX.

anche per i testi da comporrre con pdflatex. I motivi sono molteplici, alcuni giustificabili, altri difficili da spiegare, nel senso che si potrebbero riassumere nell'antico proverbio: "Chi lascia la via vecchia per la nuova, sa quel che lascia e non sa quel che trova". Non molto intelligente, ma probabilmente nessuno è indenne da questo tipo di ragionamenti. Oggi non rifarebbe più questo errore, ma userebbe direttamente lualatex.

Tra le altre cose bisogna disporre di uno *shell editor* capace di gestire la codifica `utf-8`; per molti anni chi scrive ha usato WinEdt che non era in grado di gestire questa codifica. Oggi gli *shell editor* capaci di gestire questa codifica sono numerosi; tuttavia anche con questi editor non è banale introdurre i segni non presenti in tastiera; è vero, con questa codifica si può scrivere anche in cinese, ma bisogna disporre di una adeguata interfaccia per scegliere gli ideogrammi da inserire nel testo in modo che non sia necessario ricercare l'ideogramma voluto in una enorme tabella che elenca le decine di migliaia di ideogrammi disponibili.

Questo problema è molto ridotto con gli alfabeti latini, tuttavia anche con questi alfabeti i segni non letterali o paralfabetici che potrebbe essere necessario gestire sono molto numerosi. Un breve elenco non guasta.

(a) L'apostrofo spesso è un carattere speciale, leggermente inclinato, non verticale come appare sullo schermo quando si usano i caratteri ASCII.

(b) Le virgolette alte sono spesso virgolette simmetriche (`"`), invece che asimmetriche (“ e ”); talvolta sono correttamente asimmetriche, ma sono rappresentate dagli appositi caratteri non ASCII, che generalmente non sono accettabili direttamente con un encoding come `latin1`, usato in passato da molti, invece di usare l'encoding `utf8`; esse vanno quindi convertite nelle apposite sequenze delle legature `"` e `"`, oppure sostituite con macro che svolgano correttamente il loro lavoro dietro le quinte.

(c) Non è il caso di preoccuparsi per gli apici delle note; questi vengono direttamente sostituiti e rigenerati quando viene usato il comando `\footnote`.

(d) Sono invece da gestire adeguatamente gli apici di nota eseguiti mediante caratteri paralfabetici, come l'asterisco, la spada, la doppia spada, eccetera; in questi casi conviene ricorrere ad appositi pacchetti che consentono di gestire agevolmente la composizione delle note; la documentazione fornita con il sistema TEX offre diversi pacchetti e ognuno sceglierà quello che risulta maggiormente adatto alle proprie esigenze.

(e) Talvolta è presente il segno dei gradi, °; più raramente le frazioni a barra del tipo $^1\!/_2$, $^3\!/_4$, e simili. Il segno dei gradi potrebbe risultare compatibile con l'encoding `latin1`, ma si ritiene che per "pulizia" sia meglio renderlo correttamente con la sua rappresentazione mediante

i segni disponibili con LaTeX, che dispone di `\textdegree` per la composizione testuale e `^\circ` per la matematica; si è già esposto nei capitoli precedenti come definire nuovi comandi che consentano di automatizzare la scelta fra modo testo e modo matematico; se si parla di gradi Celsius[3], °C, bisogna correttamente inserire lo spazio fine, che invece non bisogna inserire quando si parla di gradi "angolari"; si confronti l'angolo di 30° con la temperatura di 30 °C.

(f) Gli apici ª e º degli ordinali femminili e maschili vanno sostituiti anche questi con gli appositi comandi disponibili con l'opzione *italian* di *babel*: `\ap{a}` e `\ap{o}`.

(g) I deponenti (e i rari ascendenti) da comporre con un carattere non inclinato in modo matematico vanno comunque corretti tutti quanti; è difficile che chi ha scritto un documento con Word o con Writer abbia avuto l'attenzione di distinguere con il font giusto i deponenti attributivi dai pedici letterali "variabili"; questa correzione, comunque necessaria, richiede la comprensione della matematica da parte del compositore.

(h) Certamente si possono presentare altri caratteri non ASCII nei file sorgente, ma quelli più frequenti sono quelli illustrati sopra; non bisogna preoccuparsi delle lettere accentate, che a rigore non sono ASCII, perché solitamente esse non appaiono come incompatibili nella finestra di editing; se lo fossero, le parole che le contengono risulterebbero marcate come ortograficamente errate, quindi vi si può intervenire caso per caso.

A chi scrive non è mai capitato di dover intervenire per le vocali accentate della lingua italiana (salvo l'eventuale correzione degli accenti gravi o acuti per la lettera 'e' dove precise norme prescrivono l'uno o l'altro accento), ma potrebbe verificarsi in lingue diverse dall'italiano, oppure, più subdolamente, in parole straniere inserite in un contesto italiano, perché in questi casi il correttore le dà comunque ortograficamente errate (di solito i correttori ortografici sono impostati per una sola lingua alla volta) e non è immediatamente chiaro se la cosa sia solo dovuta ad una parola non appartenente alla lingua di default, oppure se sia dovuta alla presenza di una lettera con diacritici che non corrisponde all'encoding preimpostato; in questi rari casi si può procedere senza preoccuparsi dell'esito della composizione; durante la compilazione eseguita con pdflatex se il segno non ASCII non è riconosciuto dalle definizioni corrispondenti all'opzione data al pacchetto *inputenc*, viene segnalato un *warning* in corrispondenza del

[3]Qui si è barato un pochino; nel file che contiene le macro specifiche per la composizione di questo testo, si sono usati comandi diversi per la composizione testuale dei gradi Celsius, `\textcelsius`, °C, e `$^\circ\mathrm{C}$`, °C, per la matematica. Si osserva che i due segni sono leggermente diversi; negli esempi qui riportati si è sempre usato il segno matematico reso il più possibile simile a quanto si ottiene con il segno testuale.

carattere non riconosciuto ed esso viene sostituito con un punto interrogativo; di conseguenza prima di eseguire nuovamente pdflatex basta sostituire la lettera incriminata con la sua definizione esplicita mediante gli appositi comandi per gli accenti e i diacritici in particolare.

Dovendo citare la locuzione irlandese[4] *Rí Teamṙać* bisogna scrivere esplicitamente `Rí Tea\.mra\.c`, perché altrimenti i caratteri puntati vengono sostituiti con punti interrogativi (*Tea?ra?*) in quanto incompatibili con l'encoding `latin1`. Questi inconvenienti non dovrebbero manifestarsi se il copia-incolla viene fatto in un documento `.tex` in codifica *utf8*, da comporre con lualatex.

Per la conversione di questi caratteri, indipendentemente dalla capacità dell'editor di gestire caratteri diversi da quelli corrispondenti al suo encoding impostato, basta copiare una qualunque istanza del carattere da sostituire, inserirla nel campo di ricerca della finestra "cerca e sostituisci", e inserire il testo o il carattere o la macro sostitutiva nel campo di sostituzione per poi eseguire una sostituzione globale. Di solito questa operazione, da ripetere per ogni carattere non ASCII, è risolutiva, anche se non guasta controllare una seconda volta che cosa sia stato effettivamente sostituito e dove.

Nel fare queste operazioni di sostituzione e di correzione si inseriscono anche nei successivi capoversi o nelle successive sezioni del testo ricomposto tutte le informazioni di mark-up necessarie, le etichette per le informazioni incrociate, i comandi per la generazione degli indici e delle varie liste di oggetti flottanti; bisogna dunque aggiungere con il mark up LaTeX quelle informazioni che non potevano essere contenute dentro i file di puro testo o formattati secondo il formato `.docx`.

L'esecuzione del programma pdflatex non dà più problemi di quelli che si hanno nella prima esecuzione di un qualunque file sorgente; certamente compariranno delle cose non errate sintatticamente, ma che non corrispondono esattamente al quel che si desiderava comporre; certamente bisognerà agire sugli oggetti flottanti per trovare loro una posizione conveniente in relazione ai parametri di collocamento automatico di cui si è parlato negli appositi capitoli. Ma questi problemini sono quelli che si incontrano sempre con qualunque documento complesso.

Per evitare di dover ricercare grossi errori, come per esempio l'assenza di una graffa chiusa o della fine di un lungo ambiente di qualunque tipo, conviene compilare il documento ogni pochi capoversi, non necessariamente ad ogni capoverso aggiunto al file in lavorazione; ecco perché si è consigliato di convertire

[4] Questa locuzione è stata ripresa dal testo di Peter Flynn [25]. In quel testo Flynn dice anche che non esiste una chiave di tastiera che consenta di inserire una 'i' priva di puntino e che perciò bisogna usare la sequenza `\i`; questo è vero per la lingua celtica usata in Irlanda e per la lingua turca; per le 'i' accentate questo era vero fino ad una decina di anni fa, quando per tutti i diversi encoding è stato dichiarato che la macro dell'accento seguita da una 'i' "normale" (col puntino) produce direttamente e correttamente la 'i' accentata con l'accento che sostituisce il puntino. Forse non è vero con tutti i possibili accenti, ma certamente lo è per i più comuni.

un capoverso alla volta per apportare le modifiche di poco conto, mentre la compilazione ad ogni gruppo di capoversi, due o tre pagine alla volta, permette di scovare le imperfezioni di più largo respiro e di eseguire le conversioni dei caratteri non ASCII meno frequentemente che non un capoverso alla volta.

Chi scrive ha convertito un certo numero di libri in questa maniera; un paio di questi libri erano atti di congressi o di simili conferenze di argomento tecnico scientifico. Specialmente per questi ultimi, i cui contributi erano stati scritti da ogni presentatore di memorie, ma con stili tutti diversi, con abilità più o meno grandi nell'uso di Word o di Writer, con competenze di vario livello in merito alle immagini allegate ai loro lavori; con formule composte spesso in violazione di ogni possibile norma di scrittura della matematica tecnico scientifica e senza servirsi di *equation editor* validi, hanno presentato notevoli problemi; sicuramente non sarebbe stato possibile eseguire nessuna conversione in forma automatica o semiautomatica.

23.2 Conversione automatica

23.2.1 Documenti in formato `.docx`

Chi scrive ha provato ad usare programmi come rtf2latex2e; qualunque word processor, come Word o Writer può aprire un documento in formato `.docx` e salvarlo in formato `.rtf` (Rich Text Format). Il programma citato, rtf2latex2e, scaricabile dalla rete per i tre sistemi operativvi principali, sembra che consenta di trasformare qualunque documento RTF, gestendo anche le figure e trattando le eventuali formule in esso contenute come immagini. L'esperienza individuale può essere molto varia con questo programma e chi scrive non è riuscito ad ottenere granché, probabilmente per sua inesperienza.

Esiste anche un plug-in per Microsoft Word che si chiama word2tex, usabile solo su piattaforme Windows; dagli archivi ufficiali CTAN è scaricabile un programma eseguibile con lo stesso nome ma datato 1988, quindi vecchiotto, che dovrebbe fare le stesse cose. In rete c'è questo plug-in commerciale aggiornato al 2008, nella forma di uno shareware, la cui licenza singola, da pagare dopo i 30 giorni di prova, non è proprio a buon mercato, ma sembra che ne valga il costo, specialmente se si esegue la conversione Word $\longrightarrow$ LaTeX con una certa frequenza. Le prove che chi scrive ha visto già eseguite da altri[5] sembrano eccellenti; il codice LaTeX è corretto e senza sbavature, anche se in realtà si deve eseguire qualche piccolissimo intervento a mano per dare qualche piccolo ritocco; il file convertito, però, è subito compilabile e il documento prodotto, a parte i font e qualche altro piccolissimo dettaglio, è sostanzialmente identico all'originale.

Il plug-in word2tex si presenta nei menù di word come una opzione del comando `Salva con nome` dove fra gli altri formati compare anche il formato TeX (in realtà vuol dire LaTeX); se il file sorgente contiene figure o equazioni, queste

[5] Chi scrive lavora su un Mac; ma il plug-in descritto qui non è installabile nella versione di Microsoft Office per Mac.

vengono correttamente convertite in modo compatibile se sono presenti sulla macchina Windows alcune librerie dinamiche sufficientemente recenti; tuttavia per questi dettagli è meglio leggere la documentazione sul sito ufficiale `http://www.chikrii.com/`.

Invece i programmi per Linux AbiWord e Kword possono aprire un documento `.docx` e salvarlo ("Salva con nome") in formato `.tex` con il mark-up di LaTeX. È chiaro che questi programmi, essendo nativi di Linux e, specialmente il secondo, essendo legati al desktop KDE, funzionano solo su macchine Linux[6]; in ogni modo nessuno vieta di installare su una macchina Windows il programma CygWin, che serve per simulare un ambiente UNIX su una macchina Windows, e successivamente installare dentro questo ambiente di simulazione uno dei due programmi in questione. Sulle macchine Mac con il sistema operativo OS X, non ci sono particolari problemi; o si scarica l'apposito programma di installazione per Mac, oppure si scarica il file sorgente e si genera l'applicazione compilando il file sorgente; questo è uno dei vantaggi conseguenti all'adozione del sistema UNIX da parte della Apple. Quindi in sostanza i due programmi nominati sono usabili su tutti e tre i sistemi operativi maggiori.

Tuttavia... nel marzo 2010 chi scrive ha provato ad usare entrambi i programmi summenzionati su una macchina Linux dotata di sistema operativo Ubuntu 9.10.

Con AbiWord si è aperto un primo file `.docx` che il programma ha importato direttamente senza problemi; lo si è salvato in formato LaTeX; si è aperto questo nuovo file, ma la conversione non ha dato luogo ad una compilazione corretta, e l'intervento a mano per renderlo compilabile è stato decisamente pesante.

Si è aperto un altro file dello stesso formato, questa volta un poco più complesso, contenente anche l'indice generale. Di nuovo l'editing per renderlo compilabile con pdflatex ha richiesto non poco lavoro. Il motivo principale è che il file `.docx` originale era stato composto su una macchina Windows, il cui fine riga è caratterizzato da due byte; uno indica di andare a capo e l'altro di iniziare una nuova riga. Nelle macchine UNIX i file di testo hanno il fine riga marcato da un solo byte, quello di inizio di una nuova riga. Benché AbiWord mostri correttamente sullo schermo il contenuto del file `.docx`, quando lo converte in formato LaTeX si comporta come se non riconoscesse entrambi i byte di fine riga del formato "Windows", ma ne riconosce uno solo, e racchiudesse l'altro in un ambiente *flushleft*; questo succede specialmente nelle parti preliminari del documento che hanno diversi spazi verticali fra capoversi introduttivi. Certo, sapendolo, si sa che bisogna cancellare un grande numero di questi ambienti contenenti solo una riga vuota, ma richiede del tempo e molta attenzione per essere sicuri di non cancellare "troppo". Bisogna in ogni caso modificare le impostazioni di default del preambolo e molte righe di mark-up interne. Fattibile, ma faticoso.

Con Kword le cose sono simili, ma leggermente migliori; il dialogo che il programma apre per svolgere la conversione richiede diverse indicazioni che

[6]Sembra che AbiWord offra anche una versione adatta alle piattaforme Windows; per maggiori informazioni si veda nel sito Internet di AbiWord.

consentono anche di scegliere l'encoding del file sorgente. La scelta della lingua principale del documento non sembra avere effetto, mentre con AbiWord non c'è personalizzazione e la lingua di default è comunque l'inglese. Entrambi specificano il formato americano *letter* per la carta, ma è immediato modificarlo in A4.

Insomma: entrambi i programmi eseguono la conversione, ma entrambi producono in uscita dei file LaTeX che richiedono non poco lavoro per renderli compilabili.

Chi scrive ha usato anche altri programmi di cui si parla nelle FAQ britanniche, ma i risultati non sono stati per niente soddisfacenti.

Non bisogna stupirsene più di tanto; i file `.docx` non contengono nessuna informazione in merito alla struttura del documento, ma contengono informazioni in merito solamente all'aspetto visuale del testo; quindi anche se questi programmi convertono il file dal formato `.docx` al formato `.tex` con il mark-up di LaTeX, i file così prodotti richiedono non poco lavoro per produrre dei documenti composti tipograficamente in modo corretto.

Per la componente Writer di OpenOffice.org e LibreOffice esiste l'estensione Writer2LaTeX che non è installata di default, ma ogni utente di OpenOffice.org e di LibreOffice sa come reperirla dal sito ufficiale di quelle suite; scaricare questa estensione e installarla è una questione di pochi click; una volta installata l'estensione, la voce "Export" del menù "File", consente di esportare il file in formato LaTeX; una finestra di dialogo viene aperta per specificare le impostazioni di default.

Il file da convertire (chi scrive ha provato a convertire un file con indirizzi internet, liste e sezionamenti) viene convertito perfettamente e pdflatex lo compila senza errori conservando abbastanza bene l'aspetto che il testo aveva nel formato `.docx`; la lingua viene riconosciuta correttamente così come i caratteri speciali, accettati come caratteri non ASCII che vengono riconosciuti nella codifica *latin1*. Se c'è qualche modesta critica da fare è quella che viene caricato un gran numero di pacchetti esterni, molti dei quali servono solo per riconoscere caratteri che la maggior parte delle volte non sono stati effettivamente usati nel file sorgente. Inoltre l'apostrofo viene sempre convertito in una virgoletta semplice alta di chiusura, ma tutto sommato non è difficile sostituire la stringa `{\textquoteright}` in un semplice apostrofo quale quello che si ottiene dalla tastiera; questa sostituzione, però, è solo estetica, perché la sua omissione non produce danni di nessun genere, salvo rendere leggermente più faticosa la lettura del file sorgente[7]. Generalmente bisogna anche cambiare il formato della carta da `letterpaper` a `a4paper`. Per il resto questo convertitore/esportatore dal formato `.docx` al formato `.tex` sembra eccellente.

Merita aggiungere che gli sviluppatori di Writer2LaTeX stanno lavorando su una ulteriore estensione, Writer4LaTeX, che dovrebbe essere già inclusa a partire

[7] Chi scrive si è rivolto al creatore di questa estensione per segnalargli questo inconveniente; egli ha prodotto una versione "sperimentale", che lo scrivente ha avuto a disposizione prima della distribuzione ufficiale, e questo inconveniente relativo all'apostrofo non si presenta più; probabilmente quando il lettore leggerà questa nota, in rete sarà disponibile la versione ufficiale modificata.

dalla versione 1.2 di Writer2LaTeX, e che dovrebbe integrare completamente Writer con LaTeX (pdflatex) in modo da svolgere le funzioni di *shell editor* per LaTeX; insomma in un certo senso svolge le funzioni di LyX, con un suo formato di default e con la possibilità di compilare direttamente con pdflatex e di visualizzare direttamente il frutto della composizione senza uscire da Writer. Il messaggio che sul sito `http://writer2latex.sourceforge.net/index4.html` accompagna le altre informazioni su questa nuova estensione richiama ancora una volta che sia Writer sia LyX hanno delle capacità limitate (*a finite feature set*), ma che l'utente esperto può sia con LyX sia con Writer inserire codice in linguaggio TeX/LaTeX in modo da sopperire alle limitazioni del programma usato per la predisposizione del file sorgente.

Il sito Wiki di LyX, `http://wiki.lyx.org/Tools"/Word2LyXMacro` contiene alcune macro Word che permettono a Word (2007 e 2010) di "autoconvertire" un documento `.docx` in un corrispondente documento `.lyx`; questo può venire aperto con LyX ed essere salvato in formato `.tex` con il mark-up di LaTeX; chi scrive ha provato a montare il file con le definizioni delle macro Word sulla versione Mac di questo programma (Microsoft Office 2004), ma l'esperimento è fallito; benché le macro siano state salvate e correttamente individuate dal programma, esse non hanno dato luogo che a un errore grave. Tuttavia l'insuccesso è dovuto ad una incompatibilità del set-up della suite Microsoft Office per Mac con il corrispondente set-up su una macchina Windows. Infatti su questa macchina la stessa procedura di installazione (Word XP) dà luogo a macro eseguibili che convertono correttamente il file `.docx` in un file `.lyx`. Aprendo il file con LyX e salvandolo in formato `.tex` con il mark-up di LaTeX, si osserva una conversione quasi perfetta subito compilabile senza errori significativi.

Gli "errori" sono in realtà impostazioni non desiderate nel preambolo. Per esempio il pacchetto *hyperref* è caricato con l'opzione *dvipdfm*, mentre sarebbe meglio che questa opzione sia presente solo nel file da compilare con latex, non con pdflatex: basta cancellare l'opzione. Curiosamente viene impostato l'encoding del file sorgente con l'opzione *latin9*[8], ma non è un problema, se lo si desidera, cambiare l'opzione in *utf8*. Ancora: viene usata di default una classe della collezione KOMA-script, ma viene specificata come opzione globale di default la lingua inglese. Come si vede piccole cose che possono venire modificate in modo semplicissimo.

Si tenga presente che la macro Word che usa questo procedimento non converte le equazioni; queste vanno quindi reintegrate a mano. Non solo ma le figure eventualmente presenti nel file `.docx` non devono essere effettivamente incluse, ma devono essere collegate con un link, cioè vengono rese visibili nel documento originale mediante un link alla loro posizione nella cartella dove risiede il file `.docx` attraverso il comando di menù LinkToFile.

[8] L'opzione *latin9* dovrebbe consentire l'uso immediato del simbolo dell'euro, senza bisogno di ricorrere al pacchetto *textcomp*; non è però chiaro in che ordine debbano essere specificate le chiamate ai vari pacchetti, per cui si possono incontrare errori oppure l'euro viene sostituito da un punto interrogativo.

23.2.2 Documenti in formato PDF

Bisogna ancora ricordare che AbiWord e Kword possono aprire o importare documenti PDF; una volta aperti, questi documenti possono venire salvati in formato `.tex` adatti al mark-up di LaTeX. Va da sé che dai documenti PDF non si riescono a convertire adeguatamente le formule; quindi queste vanno ricomposte oppure trattate come detto qui di seguito

Uno dei vantaggi di usare un file PDF da cui partire, specialmente se l'autore ha incluso le figure partendo da file adatti, con una definizione sufficientemente elevata, è che queste figure talvolta possono venire estratte, ma che comunque possono venire ritagliate dalla pagina PDF dove compaiono; lo stesso si può fare per le formule se queste sono state composte correttamente e con i font "giusti", cioè che non stonino con i font del documento.

Può darsi che l'operazione si possa fare nativamente anche con altri programmi, ma sui Mac OS X il programma Preview (Anteprima se il sistema operativo è in italiano) permette di selezionare una parte di pagina, per esempio una figura o una formula, di copiarla nel "clipboard" e poi, agendo sulla voce del menù File: "New from clipboard", incollare quanto si era copiato in un nuovo file che può venire salvato in qualunque formato fra i tanti di cui il programma è dotato, in particolare in formato PDF, dando al file un nome adeguato; questo può poi venire incluso nel documento in lavorazione con il solito comando `\includegraphics`, già descritto nei capitoli precedenti. Il programma Preview consente di scontornare l'immagine prima di salvare il file, per cui, una volta terminata l'operazione, quanto deve venire incluso è già pronto senza bisogno di ricorrere alle operazioni di "cropping" già descritte nei capitoli precedenti.

Ovviamente questa operazione di taglia e incolla delle figure e, talvolta, delle formule, può essere eseguita anche con disegni o altre forme di illustrazioni prodotte da altri programmi. Chi scrive solitamente usa questo metodo per estrarre i diagrammi di vario genere da file prodotti con Excel; il risultato è solitamente molto buono. Anche se questa non è una operazione per convertire il formato di un documento, è pur sempre un tipo di operazione che occorre fare spesso quando si compongono documenti il cui materiale ha provenienze diverse.

Chi scrive non è al corrente di tutti i possibili programmi per aprire ed elaborare file in formato PDF con le prestazioni di Preview appena descritte; egli è al corrente del programma Skim sempre per il sistema operativo Mac OS X, che svolge le stesse funzioni di Preview ed alcune altre, molto comode, fra le quali la possibilità di essere 'sincronizzato' con lo *shell editor*. Ne esistono di sicuro anche per gli altri sistemi operativi, ma è bene che ognuno se li cerchi e se li installi sapendo bene che cosa vuole ottenere.

Una informazione interessante, però, è che dalla fine del 2009 la Sun Microsystems ha messo a disposizione per il suo StarOffice e per OpenOffice.org un programma di estensione chiamato pdfimport, da installare come tutti gli altri plug-in di estensione per quei sistemi. Con questa estensione installata sarebbe possibile importare file PDF nella sua versione Draw, la parte di OpenOffice.org destinata ai disegni, e i comandi di editing disponibili sono tali da poter introdur-

re semplici correzioni (visto che senza un programma dedicato è molto difficile – parole della Sun – apportare modifiche ai file in formato PDF). Tuttavia, una volta importato il file e introdotto l'intero documento in un file del Writer di OpenOffice.org, non dovrebbe essere molto complicato esportare il file in formato LaTeX compatibile, se è già stata installata l'estensione Writer2LaTeX.

Chi scrive non ha materialmente provato questa strada, sia perché a tutt'oggi (luglio 2010) il programma è distribuito come versione 1.0, ma forse sarebbe meglio considerarlo una versione beta avanzata; sia perché apparentemente, se la cosa funziona, ogni riga viene salvata come una scatola o riga a sé stante, quindi bisognerebbe eliminare moltissime dichiarazioni di ambiente. Tuttavia, anche se l'operazione sembra richiedere un grosso lavoro, vale la pena di tenerne conto perché in alcune circostanze potrebbe essere meglio che eseguire la conversione a mano.

23.2.3 Documenti in formato XML

Da diversi anni Microsoft Word può salvare i suoi documenti in una sua versione di formato XML (eXtended Markup Language); l'estensione dei documenti in questo formato è `.docx`; dal 2003 il programma può salvare anche in formato `.xml`, ma bisogna chiederlo esplicitamente al momento di salvare il documento.

OpenOffice.org Writer ha come formato interno standard l'XML e i suoi file salvati in questo formato hanno l'estensione `.odt` se si tratta di testo con poca o nulla formattazione, oppure in formato `.sxw` se si tratta di un documento strutturato. Siccome Writer può salvare anche nei formati della suite Microsoft Office, Writer può salvare anche nei formati `.docx` e `.xml`.

Tra i programmi già elencati bisogna ricordare che AbiWord può nativamente aprire i file XML prodotti con qualunque estensione da Microsoft Word e da OpenOffice.org Writer; quindi può di fatto aprire qualunque versione standard o meno standard di documento XML, e li può salvare in formato `.tex` con il mark-up LaTeX,

Anche Kword può aprire file con i formati XML di base o le variazioni di Microsoft Word e di OpenOffice.org Writer. Anche Kword può salvare i suoi documenti in formato `.tex` con il mark-up di LaTeX.

Il programma LyX, invece, non apre questi formati XML in modo diretto; è un peccato, perché il codice LaTeX prodotto da LyX è decisamente il migliore fra quelli ottenuti per conversione automatica.

Però, almeno sul Mac, è disponibile, o scaricabile dal sito di Microsoft, l'applicativo Open XML converter che consente di convertire i file XML in formati validi per versioni precedenti a quella del 2003 della suite Microsoft Office; tra questi, ovviamente, c'è il tradizionale formato `.doc` con il quale si può operare come descritto nei paragrafi precedenti.

I file in formato `.docx` sul Mac possono venire aperti anche con la semplice applicazione TextEdit che può salvare il file in formato `.rtf`, sul quale si può operare come descritto nei paragrafi precedenti.

Anche il potentissimo editor emacs può convertire i file XML in file adatti a LaTeX; sembra anzi che esso abbia funzionalità particolarmente avanzate; chi scrive però non ne ha esperienza diretta, ma si rifà solo a quanto Peter Flynn afferma nel suo testo [25].

Peter Flynn indica anche un'altra strada per convertire i file in formato XML nel formato adatto a LaTeX; la procedura descritta da Flynn nel suo paragrafo 10.1.1 "Getting LaTeX out of XML" è abbastanza lunga da descrivere, anche perché Flynn scrive delle macro in XLM e le usa per convertire direttamente in formato adatto a LaTeX, usando un programma chiamato Saxon, un programma Java per l'elaborazione di documenti XSLT. Secondo Flynn, questo sarebbe il metodo migliore per eseguire le conversioni, ma richiede delle capacità d'uso dei PC che chi scrive ritiene molto superiori alla media degli utenti, comunque molto superiori alle sue.

Merita ricordare il programma dblatex; si tratta di un comando senza interfaccia grafica da lanciare da linea di comando; esso apre i file in formato XML e li trasforma direttamente in file LaTeX. Esso gestisce convenientemente gli *style sheets* necessari per la conversione. Esso può venire scaricato dal sito `dblatex.sourceforge.net` e le istruzioni di lavoro sono in `http://dblatex.sourceforge.net/doc/`. Esso è fornito di comandi di pre- e post-processing che consentono di eseguire tutte le operazioni per comporre il documento o direttamente attraverso pdflatex o indirettamente passando attraverso il formato PS. Durante queste operazioni le figure, per esempio, vengono convertite nei formati richiesti da pdflatex o da latex (o meglio, da dvips). In questo modo chi converte il file XML si ritrova alla fine sia con il documento composto, sia con un sorgente LaTeX che può tranquillamente modificare per ottenere un tipo di composizione diversa. Probabilmente questo pacchetto esegue le stesse operazioni descritte da Flynn e menzionate molto brevemente poco sopra, ma le esegue in modo automatico, senza o con poco intervento da parte del compositore. È interessante notare che l'autore di dblatex ha predisposto anche un programma "gemello", dbcontext, per convertire i file XML in modo da poterli comporre con il programma context.

23.3 Documenti in collaborazione

Merita riprendere il discorso sui documenti composti da più autori. Per questo scopo sarebbe bene usare strumenti opportuni, piuttosto che fare artigianalmente tutto quanto. In ogni caso è necessario che ci sia uno dei coautori che si comporti da capogruppo e che si occupi dell'omogeneizzazione dei contributi di tutti gli altri autori; lo stile letterario e lo stile tipografico dei vari contributi dovrebbe essere identico; si può forse omogeneizzare lo stile tipografico, è più difficile farlo con lo stile letterario.

Ovviamente prima di cominciare un lavoro fra più autori è bene che il coordinatore abbia cura di concordare con gli altri autori un piano di lavoro; una scaletta o un indice provvisorio a cui i coautori dovrebbero adeguarsi affinché

il lavoro sia ordinato, privo di ripetizioni e omogeneo. Certamente un gruppo di autori che si accinga a creare un lavoro a più mani sa come scegliersi il coordinatore e come organizzarsi, quindi qui non si insiste più di tanto su questa importante fase preliminare.

Tuttavia il coordinatore deve conoscere bene i software a disposizione e deve saperli usare con competenza per ottenere un lavoro finale di qualità.

Per raggiungere questo obiettivo ci sono degli strumenti utili che possono essere usati anche per comporre un documento scritto da un solo autore, ma qui probabilmente è meglio sottolineare l'utilità nel lavoro collaborativo.

Oggi l'uso del computer semplifica il flusso di lavoro, nel senso che è facile presumere che i singoli contributi siano prodotti fin dall'inizio in formato digitale, ma ciò non significa che una volta raccolti essi siano già *pronti e a posto* per la successiva fase di composizione del documento finale. È difficile che questo accada. È più consueto, invece, che si presenti la necessità di comporre una prima bozza che servirà per valutare l'opera nella sua interezza e procedere successivamente con il completamento e l'armonizzazione di tutti i capitoli. In parte ciò è dovuto al fatto che i differenti autori lavorano, perlomeno inizialmente, ognuno per conto proprio, senza vedere cosa e come fanno gli altri, ed in parte perché un elaborato si sviluppa nel tempo, in condizioni e stati d'animo differenti e mutevoli.

Per limitare le problematiche sopra accennate il coordinatore ed i vari coautori possono avvalersi di strumenti informatici e di metodi in grado di ottimizzare e semplificare il flusso di lavoro.

Nei successivi paragrafi verranno analizzati alcuni programmi scelti principalmente in base all'esperienza di chi scrive[9]. Conseguentemente non si tratta del sistema o dei programmi migliori in senso assoluto ma di sistemi ed applicazioni adoperati e vagliati nel tempo dall'autore, spesso in conseguenza di fatiche, di difficoltà, di errori commessi, di cui si è fatto tesoro in relazione:

- alla loro versatilità per un uso collaborativo;
- alla loro capacità di gestione di testi articolati;
- all'integrazione con LaTeX;
- alla loro integrazione con sintassi (*markup*) semplici quali *multimarkdown* (MMD) e *markdown* (MD) a cui far ricorso per produrre contributi testuali da utilizzare in forme diverse: per la stampa (su pdf e su carta), per la pubblicazione sul web (html), per la preparazione di presentazioni.

La compresenza di più autori comporta almeno un paio di valutazioni preliminari, fondamentali per delineare i successivi percorsi operativi.

1. Lavorano tutti con la stessa piattaforma informatica oppure operano su piattaforme differenti?
2. Qual è il livello di competenza informatica di ciascun coautore?

È importante, ad esempio, sapere fin dall'inizio del lavoro, se tutti i collaboratori conoscono LaTeX, oppure MMD e/o MD, per definire a priori in che

[9] Questa sezione è stata gentilmente messa a disposizione da Federico Morchio.

modo verranno scritti e consegnati i file di ciascuno e che tipo di applicativi si potranno adottare per la composizione della bozza. Le differenze in termini di tempo e di efficienza possono essere significative per valutare, a priori, l'entità dell'impegno che il coordinatore dovrà affrontare e per stimare i conseguenti tempi di elaborazione del lavoro.

Nei successivi sotto-paragrafi segnaliamo alcuni software utili a partire dalla fase iniziale di produzione delle bozze preparatorie fino alla produzione del documento finito, compilato con LaTeX. Le scelte indicate, incentrate principalmente sul sistema operativo Mac OS X, non vanno interpretate come un'intrinseca indicazione di preferenza di una piattaforma informatica rispetto ad altre; vanno invece intese come una limitazione dell'autore di questo contributo che non conosce a sufficienza altri sistemi per poterne scrivere con eguale competenza.

23.3.1 GIT

Il sistema GIT si basa sul fatto di comporre i propri file dentro un ambiente di sviluppo che tiene traccia delle versioni e delle modifiche fatte; allo sviluppo di un unico file o di molti file correlati possono partecipare diverse persone che contribuiscono con le loro versioni o le loro modifiche, ma c'è una sola persona che può accettare le modifiche proposte ed integrarle nel set di file, conservando però le versioni precedenti, cosicché si possano ripristinare selettivamente parti di file precedenti o interi file.

Come si può facilmente capire, questa è esattamente la soluzione alla situazione descritta nel paragrafo precedente, ma richiede che tutti i coautori conoscano LaTeX con lo stesso livello di competenza.

Non c'è ragione di entrare qui nello specifico di questa soluzione poiché nella sezione `http://www.guitex.org/home/it/guide-tematiche` del sito GUIT è scaricabile un'ottima guida dedicata, *GIT 4 LaTeX* che spiega con chiarezza come mettere al sicuro, anche su server remoti, i file in collaborazione. Perciò si rinvia direttamente a quella guida.

23.3.2 Organizzazione minimale

Se si vuole evitare di ricorrere a software specialistici o proprietari un'organizzazione pratica e logica del lavoro di gruppo si ottiene utilizzando file di *puro testo* (`txt`) o scritti con un linguaggio di *markup* molto semplice come MMD di cui si parlerà poco più avanti.

I file di ciascun coautore vanno archiviati in una cartella, eventualmente e condivisi sul web. Il coordinatore provvede ad assemblare i vari contributi secondo l'indice prestabilito.

L'uso di markup semplici esalta la versatilità del sistema; attraverso opportuni convertitori è possibile ottenere dall'unica fonte così realizzata, e senza ulteriori rifacimenti, molteplici tipologie di documenti e differenti formattazioni, per esempio file in formato *pdf*, *html*, *opml*, *tex*, *rtf*, *docx*, fornendo nel contempo al lavoro una rifinitura accurata in ottica compositiva. Per approfondire questi

aspetti si veda l'interessante articolo [19] sull'uso di Pandoc, di cui si è già parlato in questo capitolo; questo programma è definito dal suo autore come lo strumento più generale per la conversione di file da un formato a un altro.

23.3.2.1 File `txt`

I file `txt`, se inseriti in strutture gerarchiche bene organizzate, rappresentano un modo sicuro e versatile per costruire e gestire un documento complesso. L'organizzazione è fondamentale sia in relazione alla struttura delle cartelle sia in relazione ai nomi da assegnare ai file.

Questi file sono immuni dalle differenze di piattaforma, resteranno sempre accessibili, purché si usino codifiche standard ben definite. La probabilità che restino accessibili per sempre è molto più alta rispetto a qualsiasi altro formato di file di testo attualmente in circolazione. Inoltre essi sono leggeri e non necessitano di software particolari e costosi per essere letti e modificati.

Ovviamente immagini e tabelle (non gestite dal formato) andranno conservate in cartelle apposite, ordinatamente organizzate e riferite ai testi ed alle sezioni che le dovranno contenere.

Questi file andranno successivamente assemblati e composti (dal coordinatore dell'opera) con altri strumenti software da scegliere in funzione del sistema di lavoro adoperato per la produzione del documento finale.

23.3.2.2 File mmd e md

Per i file creati usando i markup MMD e MD valgono le stesse considerazioni per la gestione generale descritte per i file `txt` perché sono a tutti gli effetti file di puro testo

Ricordiamo che il markup *multimarkdown* è una evoluzione di quello denominato *markdown*; il primo possiede peculiarità più specificatamente rivolte alla produzione di testi in formato PDF, mentre il secondo è finalizzato al formato *html*. Nei rispettivi siti si legge quanto segue.

> **Markdown**, or MD, is a text-to-HTML conversion tool for web writers. Markdown allows you to write using an easy-to-read, easy-to-write plain text format, then convert it to structurally valid XHTML (or HTML). Thus, Markdown is two things: (1) a plain text formatting syntax; and (2) a software tool, written in Perl, that converts the plain text formatting to HTML.
>
> **MultiMarkdown**, or MMD, is a tool to help turn minimally marked-up plain text into well formatted documents, including HTML, PDF (by way of LaTeX), OPML, or OpenDocument (specifically, Flat OpenDocument (`.fodt`), which can in turn be converted into RTF, Microsoft Word, or virtually any other word-processing format). MMD is a superset of the Markdown syntax, originally created by John Gruber. It adds multiple syntax features (tables, footnotes, and citations, to name a few), in addition to the various output formats listed above (Markdown only creates HTML).

> Additionally, it builds in "smart" typography for various languages (proper left- and right-sided quotes, for example).

Con le loro sintassi specifiche *multimarkdown* o *markdown*, pur trattandosi di veri e propri linguaggi informatici, rendono questi file facilmente intelligibili al cervello umano, molto più facilmente rispetto al caso che fossero scritti col markup *html*.

Per maggiori informazioni su questi linguaggi di markup si veda in:

`http://fletcherpenney.net/multimarkdown/`
e
`http://daringfireball.net/projects/markdown/`.

Questi file, se aperti con appositi software, possono essere visualizzati anche in modo sofisticato, cioè permettono di vedere il testo con formattazioni, immagini, tabelle, note a piè di pagina, eccetera.

Su Mac sono disponibili molte applicazioni che ne consentono la gestione. Tra queste ve ne sono due particolarmente interessanti nell'ottica di questa guida. Si tratta dei programmi Marked2 (`http://marked2app.com`) e MultimarkdownComposer (`http://multimarkdown.com`).

Il primo programma consente di visualizzare un file (che può anche essere salvato in formato `txt`) usando *temi* pre-impostati o di propria creazione, o di esportarlo in diversi formati alternativi: *html, pdf, rtf, rtfd, docx, odt, md*. La sua utilità si esplica, ad esempio, nelle fasi iniziali di un'opera di gruppo, sulle bozze: consente di lavorare su file di puro testo senza perdere la possibilità di conferire loro un aspetto formattato (più o meno minimale) che ne facilita ulteriormente la leggibilità. Il file deve essere scritto con un altro editor, un qualsiasi editor di testi, dal più sofisticato al più spartano, e salvato in MMD, oppure MD, oppure `txt`. L'uso di MMD consente di attribuire a ciascun file un insieme di *metadati*, fattore che, in molti casi, riveste una significativa utilità pratica e gestionale.

Il secondo programma MultimarkdownComposer dispone anche di un proprio editor interno e offre anche la possibilità di esportare in formato `tex`; è un programma commerciale che si può usare solo su piattaforme Mac al costo di una decina di euro.

Attraverso l'accurata predisposizione di un albero di cartelle e file è possibile, già a questo punto, ottenere ottimi risultati sia per lunghi documenti articolati sia per lavori meno impegnativi ma al tempo stesso strutturati, ad esempio un articolo.

23.3.3 Andare oltre: Scrivener

Il passo successivo, nel flusso metodologico della preparazione di un documento collettivo, è rappresentato dall'adozione di un *unico* software con cui:

- gestire ed organizzare tutti i file prodotti dai vari coautori (eventualmente con uno dei sistemi appena visti);

- organizzarli in parti, capitoli, sezioni, sottosezioni eccetera;
- modificare in corso d'opera l'indice generale dell'opera stessa.

Si tenga conto che è impossibile sintetizzare in poche pagine le funzioni disponibili in un software strutturato e versatile come Scrivener. In questa sede analizzeremo solo le caratteristiche legate ai temi di questa guida, rimandando chi desideri un approfondimento al sito web `https://www.literatureandlatte.com` e ad un *e-book* dedicato, reperibile nel sito `http://www.takecontrolbooks.com/scrivener-2`.

Scrivener è un applicativo in grado di amministrare all'interno di un unico file con estensione `.scriv` i molteplici contribuiti che danno luogo ad un'opera testuale; è uno strumento capace di gestire:

- le porzioni testo, le immagini, le tabelle, i weblink, le citazioni, il materiale di ricerca, eccetera, ivi compresi tutti i contributi inseriti adottando la sintassi MMD;
- le annotazioni correlate a ciascuno di essi;
- le annotazioni relative all'intera opera;
- le etichette descrittive associate a ciascuna porzione di testo che ne descrivono:
  - la tipologia gerarchica rispetto al sezionamento (parte, capitolo, paragrafo, sottoparagrafo e così via);
  - lo stato di avanzamento (To-do, first-draft, final-draft, o altro secondo le proprie necessità personali);
- gli elementi di studio e ricerca necessari ma che non entreranno a far parte della compilazione finale del documento.

Si tratta di un programma shareware, dal costo abbordabile di poche decine di euro, da pagare dopo il trentesimo giorno di uso effettivo; il denaro speso è ben ripagato dai vantaggi ricavabili con l'uso del programma.

Nasce come applicazione per *Mac OS X*, progettata da uno scrittore e specificatamente dedicata all'uso di chi deve scrivere libri, rapporti, manuali, articoli, copioni, eccetera. Grazie alla sua versatilità nel tempo ha acquisito una buona quota di mercato e si è aperta anche alle piattaforme Windows e Linux.

Per questa guida gli aspetti di interesse sono rappresentati dalle capacità di:

- gestire file di testo provenienti da altre applicazioni (ad esempio Word, Writer, Pages,...),
- gestire il *multimarkdown*,
- stampare in formato PDF il testo marcato in *multimarkdown* attraverso una compilazione che si appoggia a LaTeX,
- generare file `tex` originati da testi marcati in *multimarkdown*.

In Scrivener un documento è suddiviso in tanti frammenti di testo. Ogni frammento di testo è un file; ciò è utile per il lavoro in senso generale e fondamentale nel caso in cui le bozze provengano da file di testo redatti ed archiviati

nei modi descritti in precedenza. Questi frammenti di testo prendono il nome di *scrivening*. Essi vengono ordinati nel raccoglitore (*binder*) dall'autore o, nel caso di più autori, dal coordinatore. Il *binder* è il gestore del documento, contiene e consente l'ordinamento dei vari scrivening, compresi quei file di lavoro che non entreranno a far parte del documento finale compilato (vedi figura 23.1).

Rispetto all'uso di comuni text editor e word processor, come per esempio Word, Pages, TextEdit, Writer delle suite Libre Office o OpenOffice, ciò agevola la gestione di un testo lungo ed articolato da sezionare in parti, capitoli e paragrafi, numerati e riferiti ad un indice generale unico, e facilita l'elaborazione dei contenuti in modo selettivo sempre lavorando all'interno di un unico file `.scriv`. In questo modo si può lavorare con grande comodità pratica di gestione del file a capitoli e paragrafi non necessariamente consecutivi e/o in sequenza.

Una funzione molto efficace, specie in caso di lavoro collaborativo e multipiattaforma, consente di importare un file di Word dividendolo, se serve, in tanti scrivening. Ciò, però, avviene solo se chi lo ha redatto ha avuto l'accortezza di anteporre uno o più segni `#` al titolo di ciascun capitolo o paragrafo. Questo aspetto evidenzia, fra l'altro, la necessità di redigere preventivamente le linee guida operative di lavoro comune.

Con eguale versatilità si possono trascinare direttamente nel *binder* i file testuali dei vari coautori. In questo caso ogni documento trascinato rappresenterà uno scrivening.

Gli scrivening risiedono nelle cartelle del binder; queste possono essere inserite una dentro l'altra a formare una specie di albero strutturato. In tal caso si genereranno automaticamente, in fase di compilazione, i capitoli, i paragrafi e i sottoparagrafi. Nella stampa da MMD in PDF e nell'esportazione in LaTeX si originano così i necessari comandi `\chapter`, `\section`, `\subsection`, `\subsubsection`, eccetera.

Nella versione per Mac sono disponibili altre funzionalità che verrano descritte nelle sezioni seguenti.

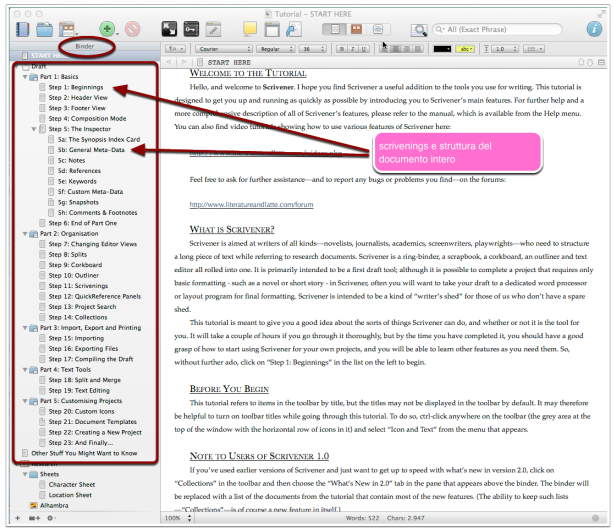
23.3.3.1 Compilazione del documento finale in PDF

Dal punto di vista operativo la situazione ottimale si presenta quando tutti i contributi testuali sono redatti fin dall'origine con sintassi MMD. Altrimenti, per poter usufruire della caratteristiche descritte nei successivi paragrafi, toccherà al coordinatore provvedere all'applicazione del *multimarkdown* agli scrivening che ne sono sprovvisti.

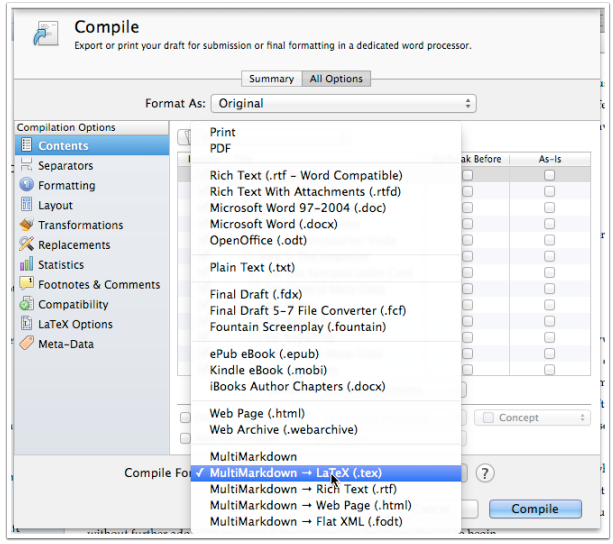
Il documento finale, costituito dagli *scrivenings* scritti in MMD e raccolti nel binder, va compilato in Scrivener per poter essere stampato o esportato in altri formati; qui ci limitiamo alle formattazioni che afferiscono al mondo LaTeX (vedi figura 23.1 b).

Possiamo percorrere due vie.

1. Compiliamo direttamente da MMD per ottenere un documento in PDF;
2. Compiliamo da MMD generando un file con estensione `tex` da gestire con applicativi specifici per LaTeX.



(a) la finestra principale di scrivener



(b) I formati di esportazione

Figura 23.1: La finestra di compilazione di Scrivener

Avendo deciso di *stampare* direttamente da MMD in PDF selezioniamo la voce:

```
multimarkdown --> pdf
```

come mostrato nella figura 23.3a. Scrivener eseguirà tre compilazioni tramite LaTeX per costruire indice, glossario, bibliografie, eccetera, restituendo un documento PDF formattato secondo le scelte impostate nella finestra 'LaTeX Options': *memoir* (book), *article*, oppure *Custom*. Quest'ultima opzione presuppone che i tre campi: 'Header', 'Begin Document' e 'Footer' vengano riempiti con le informazioni consuete necessarie per lavorare in LaTeX (vedi figure 23.1 e 23.2a).

Per l'uso pratico del sistema di lavoro, il documento composto in PDF da Scrivener, via LaTeX, è comodo:

- nelle fasi intermedie di lavoro, per la revisione delle bozze (che così avranno già una prima impaginazione in stile LaTeX);
- per chi non conosce (o conosce ancora poco) LaTeX ma desidera sfruttarne alcune potenzialità.

23.3.3.2 Compilazione del documento finale in file `tex`

L'alternativa all'output diretto in PDF è fornita dalla compilazione in uno o più file `tex` da gestire successivamente con un applicativo specifico per LaTeX (se ne veda la pagina di impostazione nella figura 23.3 b). Compilare i vari scrivening in tanti file `tex` diventa efficace se si prevede di organizzare il lavoro in LaTeX adoperando `\include` e/o `\input`. Il file ottenuto da Scrivener è già compilabile (da LaTeX), così com'è, oppure può essere *manipolato* impostando da capo la classe del documento, il preambolo e i pacchetti in relazione alle capacità ed alle necessità. La derivazione diretta dalla compilazione via Scrivener dà origine ad una *grammatica* LaTeX non sempre ottimale, sicché non guasta (sebbene non sia sempre necessario) procedere a un'attenta revisione preventiva per rifinire il codice.

23.3.4 Commenti in stile *html*

Lavorando all'interno di Scrivener con markup MMD è possibile servirsi di sintassi e comandi abituali di LaTeX. Per farlo si ricorre ai *tag* del tipo `<!-- testo -->` secondo la sintassi tipica adoperata per annotare il codice o la pagina scritti in *html*.

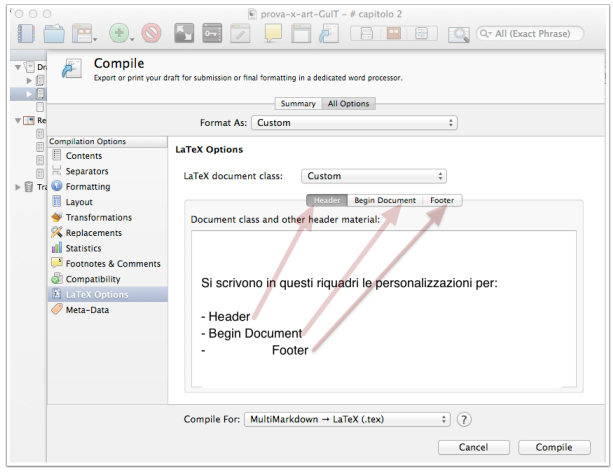
Supponiamo, ad esempio, di voler apporre una nota a margine del testo. All'interno dello scrivening, in corrispondenza del punto in cui va inserita la nota, scriviamo il codice LaTeX come se fosse un commento *html*, così:

```
<!--\marginpar{testo della nota a margine}-->
```

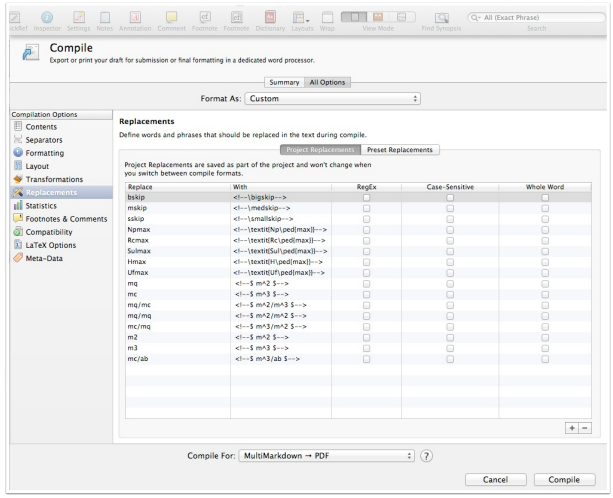
Al termine della compilazione lanciata da Scrivener comparirà la nota nel punto voluto[10]. Con questo semplice metodo si possono includere molti tratti di codice

Nota a margine del testo predisposta in Scrivener.

[10] La nota a margine sarà stampata nel PDF in caso di compilazione diretta. Se invece si

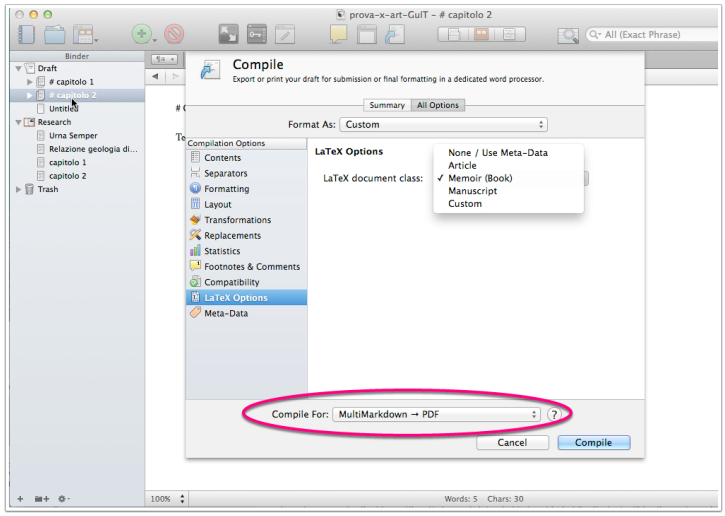


(a) personalizzare il documento finale con LaTeX

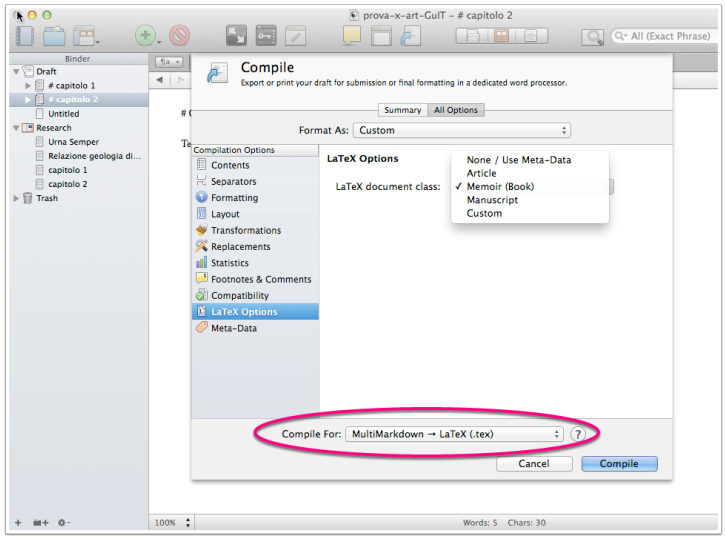


(b) uso di 'Replacement' insieme ai commenti *html*

Figura 23.2: La compilazione da Scrivener



(a) compila da MMD a PDF



(b) compila da MMD a LaTeX

Figura 23.3: La compilazione da Scrivener in MMD o LaTeX

LaTeX: scatole di testo, anche lunghe, o altre specifiche azioni quali, ad esempio, formule matematiche o testi particolari.

Un altro aspetto che merita di essere segnalato e molto utile sul piano pratico è dato dall'utilizzo congiunto dei commenti *html* e della funzione 'Replacements' che appare tra le opzioni impostabili nei dialoghi della compilazione (vedi figura 23.2b). Si possono impostare sostituzioni per semplificare la digitazione dei testi in *mmd*. Ipotizzando, ad esempio, di dover scrivere molte unità di misura secondo le prescrizioni del *Sistema Internazionale*, può diventare fastidioso dover ricorrere spesso al tag *html*. È più semplice impostare la sostituzione di un testo *abbreviato* e comodo da digitare (supponiamo *mq*) via 'Replacements'. Per farlo, in corrispondenza della colonna 'Replace' scriviamo il testo abbreviato ('mq' oppure 'm2'), e nella colonna 'with' immettiamo il testo corretto racchiuso nel commento *html*, che in questo caso potrà essere:

```
<!-- \ensuremath{\mathrm{m^2}} -->
```

In tutto il documento potremo scrivere l'abbreviazione *mq* ogni volta che ci servirà sapendo che nel documento stampato (PDF) o nel file `tex` l'unità di misura apparirà scritta nel modo corretto (m^2), con l'esponente 2. Con un po' di esercizio si possono comporre sintassi molto più complicate.

Questo metodo è utilizzabile con profitto, per esempio, per mettere in posizione i comandi di spaziatura verticale tra i capoversi. In questo modo digitando, in Scrivener, le lettere *bskip* ed avendo impostato nel 'Replacement' che ad esse corrisponda `<!-- \bigskip -->`, si ottiene:

- nel testo stampato lo spazio maggiore desiderato;
- nel nel file `tex` l'inserimento del comando `\bigskip`.

Come molte applicazioni complesse e versatili l'uso genera la crescente conoscenza delle potenzialità e congiuntamente la possibilità di sviluppare usi particolarmente adatti alle proprie necessità.

In ogni caso, Scrivener non può essere visto o considerato come uno *shell editor* per LaTeX. La distanza da TeXShop, TeXworks, Texpad (per citare solo alcuni applicativi specialistici) è molta ed incolmabile. Tuttavia rappresenta uno strumento valido per avvicinarsi *in punta di piedi* a questo mondo. Per ottenere in breve tempo risultati soddisfacenti è sufficiente imparare *multimarkdown*, impegno decisamente più semplice e meno dispendioso in termini di tempo e fatica. Certo, iniziando con meno fatica, rispetto all'approccio diretto a LaTeX, la voglia di migliorare, visti da vicino i risultati, non potrà fare altro che spingere molti utenti verso la ricerca di approfondimento e quindi verso l'apprendimento di LaTeX e dei suoi strumenti specifici.

Sul forum di Scrivener, nella categoria 'multimarkdown', si trovano spunti ed esperienze interessanti. Tra questi una guida per principianti (in inglese), dal titolo *A toad's guide to using Scrivener, MultiMarkDown and LaTeX*, scaricabile all'indirizzo web `http://www.literatureandlatte.com/forum/viewtopic.php?f=21&t=17239&start=0`.

esporta in un file *tex*, il codice LaTeX sarà posto correttamente nel punto desiderato.

23.4 Conclusioni

Come si vede i processi per importare in documenti LaTeX il contenuto di documenti in formati diversi sono piuttosto complessi, non tanto perché ci siano delle operazioni difficili da eseguire, quanto perché a tutt'oggi non esiste un modo solo, unificato e corretto per mettere in un documento LaTeX delle informazioni tratte da altri documenti che sono privi di tutte le informazioni necessarie. Nessun programma, al di là di alcune impostazioni di default, può inventarsi che cosa aveva in mente chi ha scritto originariamente il documento iniziale.

Va aggiunto che gli utenti dei word processor di solito non usano quel minimo di mark-up che programmi come Microsoft Word o OpenOffice.org Writer consentono, per esempio di scegliere gli header dall'apposito menù. Quegli header sono necessari per eseguire automaticamente l'indice del documento, ma la maggior parte dei documenti sono privi di indice e gli utenti preferiscono scrivere l'intestazione e poi cambiare il font e la serie con semplici click del mouse, piuttosto che cliccare su un menù a discesa che offre una vasta scelta di header ma con poche spiegazioni di che cosa significhino quei termini. Risultato: anche il miglior convertitore fa fatica a convertire in formato adatto a LaTeX, per cui il compositore deve consumare un tempo notevole per correggere quello che il convertitore non ha saputo fare adeguatamente per trasformare un documento di scarsa qualità in un documento che vorrebbe essere di qualità assai superiore, ma non può proprio per mancanza di informazione adeguata.

Quando, quindi, bisogna produrre un documento di qualità con LaTeX partendo da numerosi contributi scritti con word processor diversi, il compositore deve armarsi di santa pazienza anche se ricorre a programmi di conversione che, però, ricordiamolo bene, non possono fare l'impossibile.

Capitolo 24

Simbologia matematica e fisica

24.1 Unità di misura del Sistema Internazionale

Le unità fondamentali del Sistema Internazionale sono raccolte nella tabella 24.1; secondo le norme internazionali gli angoli piani e solidi sono considerati "quantità derivate adimensionate"; infatti le norme affermano che "le unità *radiante* e *steradiante* devono essere considerate come unità derivate adimensionate che possono essere usate od omesse nelle espressioni delle unità derivate". È per questo che più avanti per alcune grandezze fisiche saranno indicate *fra parentesi* le unità di misura contenenti anche i radianti o gli steradianti in quei casi in cui il loro uso consente di distinguere specie fisiche diverse ma apparentemente equidimensionate.

Si veda inoltre l'avvertenza nella pagina 634.

Grandezza fisica	Unità	Simbolo
lunghezza	metro	m
massa	kilogrammo	kg
tempo	secondo	s
corrente elettrica	ampere	A
temperatura termodinamica	kelvin	K
quantità di sostanza	mole	mol
intensità luminosa	candela	cd

Tabella 24.1: Unità fondamentali

Tutte queste unità, nonché quelle delle tabelle successive, possono essere precedute dai prefissi decimali raccolti nella tabella 24.2. Si ricorda che i prefissi vanno usati isolatamente (in passato non era infrequente osservare il prefisso

millimicro al posto del prefisso corretto *nano*). Quando l'unità di misura con prefisso è elevata ad un esponente, questo si intende applicato all'unità completa di prefisso: $3\,\mathrm{cm}^3$ indica un volume di $3\,(10^{-2}\mathrm{m})^3 = 3\times10^{-6}\,\mathrm{m}^3$ e non un volume di $3\times10^{-2}\,\mathrm{m}^3$.

Quando si parla di bit e di byte[1] si usano i prefissi binari; questi sono riportati nella tabella 24.3; essi sono ufficiali e prescritti dalle norme ISO dal 1998 e dovrebbero venire sempre usati quando si parla di quantità "informatiche", come le capienze dei dischi, o le dimensioni di certe memorie, o le velocità di trasmissione, e simili. Si noti che per l'unità kibi (kilo binario) il simbolo comincia con una K maiuscola; non è un errore: tutti i prefissi binari cominciano con lettere maiuscole incluso quello che ricorda il prefisso kilo; lo si ricorda solamente, ma Ki sta per 1024, mentre k sta per 1000; sono evidentemente due cose diverse.

Per quanto riguarda le cosiddette unità logaritmiche, sono codificate quelle della tabella 24.4; si ricorda che i nomi che vengono dati a queste unità servono solo a ricordare quale base è stata usata per il calcolo del logaritmo. Si richiama l'attenzione anche sulla corretta scrittura dei simboli dB e Np, che invece si vedono così spesso scritti in modo errato. Per quanto riguarda le unità di attenuazione e di guadagno si usano i logaritmi decimali per i decibel (vedi la nota 1), o naturali per i neper, ed in più si hanno definizioni diverse a seconda che il rapporto di cui si calcola il logaritmo sia eseguito fra grandezze di potenza o energia, oppure fra grandezze di campo:

$$\alpha = 10\log_{10}\frac{P_1}{P_2} \qquad \text{oppure} \qquad \alpha = \frac{1}{2}\log_{\mathrm{e}}\frac{P_1}{P_2}$$

dove P_1 e P_2 sono potenze, oppure

$$\alpha = 20\log_{10}\frac{V_1}{V_2} \qquad \text{oppure} \qquad \alpha = \log_{\mathrm{e}}\frac{V_1}{V_2}$$

dove V_1 e V_2 sono tensioni.

Per gli intervalli di frequenza si usano i logaritmi in base 2 per le ottave, o in base 10 per le decadi

$$I = \log_2\frac{f_2}{f_1} \qquad \text{oppure} \qquad I = \log_{10}\frac{f_2}{f_1}$$

È stato necessario introdurre molte altre unità per le grandezze fisiche derivate, al fine di evitare di dover usare lunghi elenchi di unità fondamentali elevate a potenze insolite, che sarebbe fra l'altro troppo complicato ricordare; queste unità derivate sono elencate nella tabella 24.5.

Un cenno particolare merita il *litro* perché sono leciti ben due simboli per questa unità: l, L; viene usato (impropriamente) anche ℓ, ma non è 'legale'; invece

[1] Il bit e il byte non hanno simboli ufficiali per indicarli; bisognerebbe sempre usare i nomi per disteso; è consuetudine di usare il B per il byte e il b per il bit; tuttavia il B per il byte collide con il B per il bell. Bisogna stare attenti, quindi a non mescolare i significati diversi dello stesso simbolo in contesti dove il senso del discorso non permette di disambiguare il significato del simbolo.

Prefisso	Valore	Simbolo	Prefisso	Valore	Simbolo
quetta	10^{30}	Q	quecto	10^{-30}	q
ronna	10^{27}	R	ronto	10^{-27}	r
yotta	10^{24}	Y	yocto	10^{-24}	y
zetta	10^{21}	Z	zepto	10^{-21}	z
exa	10^{18}	E	atto	10^{-18}	a
peta	10^{15}	P	femto	10^{-15}	f
tera	10^{12}	T	pico	10^{-12}	p
giga	10^{9}	G	nano	10^{-9}	n
mega	10^{6}	M	micro	10^{-6}	µ
kilo	10^{3}	k	milli	10^{-3}	m
etto	10^{2}	h	centi	10^{-2}	c
deca	10^{1}	da	deci	10^{-1}	d

Tabella 24.2: Prefissi decimali. I prefissi indicati nelle prime due righe sono stati introdotti dal CIPM nel 2022.

Prefisso	Valore	Simbolo
kibi	2^{10}	Ki
mibi	2^{20}	Mi
gibi	2^{30}	Gi
tebi	2^{40}	Ti
pebi	2^{50}	Pi
exbi	2^{60}	Ei

Tabella 24.3: Prefissi binari

Grandezza	Unità	Simbolo
attenuazione, guadagno	decibel	dB
attenuazione, guadagno	neper	Np
intervallo di frequenza	ottava	ott
intervallo di frequenza	decade	dec

Tabella 24.4: Unità logaritmiche

Grandezza fisica	Unità	Simbolo
angolo piano	radiante	rad
angolo solido	steradiante	sr
frequenza	hertz	Hz
forza	newton	N
pressione	pascal	Pa
lavoro, energia	joule	J
potenza	watt	W
carica elettrica	coulomb	C
tensione elettrica	volt	V
capacità elettrica	farad	F
resistenza elettrica	ohm	Ω
conduttanza elettrica	siemens	S
flusso di induzione magnetica	weber	Wb
induzione magnetica	tesla	T
induttanza	henry	H
flusso luminoso	lumen	lm
illuminamento	lux	lx
attività di un radionuclide	becquerel	Bq
dose assorbita	gray	Gy
equivalente di dose	sievert	Sv
attività catalitica	katal	kat

Tabella 24.5: Unità derivate

questo simbolo, se fosse legale, costituirebbe una scelta quanto mai opportuna, perché evita ogni possibile confusione della "l" minuscola con la cifra "1", e quella della "L" maiuscola con la cifra "4' (non a stampa, ma come si scrive questa cifra negli Stati Uniti).

Le unità ammesse sono riportate nella tabella 24.6.

Infine sono ancora *tollerate* o *temporaneamente accettate* alcune altre unità *in via di estinzione*; quelle che si sono già estinte (come l'atmosfera, il quintale, il millimetro di mercurio — ammesso solo in campo medico —, il poise, eccetera) non sono nemmeno elencate proprio per evitare che possa venire la tentazione di usarle ancora. Le convenzioni internazionali, a cui l'Italia aderisce, faranno sparire in un prossimo futuro anche queste unità tollerate, che sono elencate nella tabella 24.7.

Nelle tabelle 24.1–24.7 si notano delle assenze vistose, oltre a quelle già segnalate; in particolare mancano tutte le unità CGS, dagli erg alle dine, dai gauss agli oersted, tanto per citare quelle più comuni; non sono accettati il kilogrammo-forza, il micron, il carato metrico, la caloria, l'atmosfera. Si notano inoltre le assenze dei simboli cc, mc, mmc, mq, che sono scorrettamente tanto comuni in alcune scienze; al loro posto vanno usati i simboli corretti cm^3, m^3, mm^3, m^2. Una volta, ai tempi della dattilografia, era piuttosto laborioso inserire

Grandezza fisica	Unità	Simbolo	Equivalenza
angolo piano	grado sessagesimale	°	$1° = \pi\,\mathrm{rad}/180$
angolo piano	minuto sessagesimale	′	$1′ = \pi\,\mathrm{rad}/10\,800$
angolo piano	secondo sessagesimale	″	$1″ = \pi\,\mathrm{rad}/648\,000$
angolo piano	gon o grado centesimale	gon	$1\,\mathrm{gon} = \pi\,\mathrm{rad}/200$
angolo piano	giro	giro	$1\,\mathrm{giro} = 2\pi\,\mathrm{rad}$
area	ara	a	$1\,\mathrm{a} = 100\,\mathrm{m}^2$
area	ettaro	ha	$1\,\mathrm{ha} = 10\,000\,\mathrm{m}^2$
volume	litro	l, L	$1\,\mathrm{L} = 1\,\mathrm{dm}^3$
tempo	minuto	min	$1\,\mathrm{min} = 60\,\mathrm{s}$
tempo	ora	h	$1\,\mathrm{h} = 3600\,\mathrm{s}$
tempo	giorno	d	$1\,\mathrm{d} = 86\,400\,\mathrm{s}$
massa	tonnellata	t	$1\,\mathrm{t} = 1000\,\mathrm{kg}$
massa	unità di massa atomica	u	$1\,\mathrm{u} = 1{,}660\,57 \times 10^{-27}\,\mathrm{kg}$
lavoro, energia	elettronvolt	eV	$1\,\mathrm{eV} = 1{,}602\,19 \times 10^{-19}\,\mathrm{J}$
lavoro, energia	kilowattora	kW h	$1\,\mathrm{kW\,h} = 3{,}6\,\mathrm{MJ}$
carica elettrica	amperora	A h	$1\,\mathrm{A\,h} = 3600\,\mathrm{C}$
temperatura Celsius	grado Celsius	°C	$1\,°\mathrm{C} = 1\,\mathrm{K}$ ma differisce lo zero della scala: $t = T - 273{,}15\,\mathrm{K}$

Tabella 24.6: Unità di misura legalmente ammesse

Grandezza fisica	Unità	Simbolo	Equivalenza
lunghezza	miglio marino	miglio marino	$1\,\text{miglio marino} = 1852\,\text{m}$
lunghezza	ångström	Å	$1\text{Å} = 10^{-10}\,\text{m}$
area	ara	a	$1\,\text{a} = 100\,\text{m}^2$
area	ettaro	ha	$1\,\text{ha} = 10\,000\,\text{m}^2$
area	barn	b	$1\,\text{barn} = 10^{-28}\,\text{m}^2$
pressione	bar	bar	$1\,\text{bar} = 100\,\text{kPa}$
velocità	nodo	nodo	$1\,\text{nodo} = (4{,}63/9)\,\text{m/s}$ $1\,\text{nodo} = 1\,\text{miglio marino/h}$
accelerazione	gal	Gal	$1\,\text{Gal} = 1\,\text{cm/s}^2$
attività di radionuclide	curie	Ci	$1\,\text{Ci} = 3{,}7 \times 10^{10}\,\text{Bq}$
esposizione a raggi X o γ	roentgen	R	$1\,\text{R} = 2{,}58 \times 10^{-4}\,\text{C/kg}$
dose assorbita	rad	rad, rd	$1\,\text{rd} = 1\,\text{cGy}$
equivalente di dose	rem	rem	$1\,\text{rem} = 1\,\text{cSv}$

Tabella 24.7: Unità di misura temporaneamente accettate

gli esponenti; oggi con i sistemi di videoscrittura, in particolare di composizione tipografica, come LaTeX, gli esponenti non sono più un problema, quindi quelle abbreviazioni errate, tollerate per necessità di cose fino agli anni '80, oggi non sono più accettabili.

24.2 Simboli matematici nelle scienze

In questo paragrafo sono raccolti i simboli matematici più comuni che si impiegano nelle scienze e nella fisica; essi sono ispirati tra l'altro alle norme CNR-UNI 80000.

Come al solito l'elenco non è e non può essere completo, ma può servire da guida o modello per preparare un analogo elenco qualora si facesse uso di una simbologia fisico-matematica piuttosto elaborata.

Nella tabella 24.8 le lettere a e b sono due numeri reali qualsiasi, i, j, k, n sono numeri interi, z, s sono variabili o numeri complessi, x,y, (talvolta anche z), e t sono variabili reali, D è un dominio, A, B, C, P sono punti del piano o dello spazio. La colonna intestata **Simbolo** contiene il segno grafico del simbolo, oppure un'espressione che ne fa uso.

24.3 Nomenclatura

È praticamente impossibile fare un elenco di tutti nomi delle grandezze che vengono usate in ogni scienza, dalla fisica alla medicina, dall'elettronica alla

geologia. Si ritiene però cosa utile riprendere l'elenco del prospetto IV della norma CNR-UNI 80000, ampliandolo un poco e aggiungendovi il simbolo (o una scelta di simboli) che sono comunemente accettati in ogni scritto scientifico, senza che sorga la necessità di compilare un elenco delle grandezze e dei simboli usati.

Nel compilare la tabella 24.9 si è tratta ispirazione dalle norme CNR-UNI, dalle norme CEI, dal fascicolo CEI di nomenclatura nucleare (tutti integrati nella norma ISO 80000), dal documento sulla nomenclatura pubblicato dalla Società Internazionale di Fisica, senza inventare nulla, ma operando solo delle scelte fra le grandezze o i simboli che sono stati inclusi o esclusi nella tabella.

Fra parentesi, nella colonna delle unità di misura, vi sono delle indicazioni ulteriori che comprendono anche i radianti o altre unità come i neper o i cicli, quando è parso che la specificazione di queste unità accessorie rendesse più chiara la differenza fra grandezze di specie diversa ma apparentemente equidimensionate.

La tabella 24.9, come detto sopra, è certamente incompleta, ma rappresenta comunque un modello da imitare qualora fosse necessario fare un elenco delle grandezze e dei simboli usati nel documento.

Tabella 24.8: Simboli matematici

Simbolo	**Significato**	**Note**
,	virgola decimale	Non usare il punto per separare la parte intera dalla parte decimale. Non usare nemmeno altri separatori tra i gruppi di tre cifre prima e dopo la virgola. Il punto decimale si può usare solo scrivendo in inglese, ma i separatori fra gruppi di tre cifre devono essere assenti lo stesso. In entrambi i casi il separatore può essere costituito da uno spazio fine non separabile
∞	infinito	
π	$\pi = 3{,}141\,592\ldots$	Se il font lo consente, sarebbe desiderabile stampare la costante matematica π con un carattere non inclinato: $\mathrm{\pi}$.

continua

Continua tabella 24.8

Simbolo	Significato	Note
e	$\mathrm{e} = 2{,}718\,281\ldots$	Va scritta in tondo, se non altro per distinguerla dalla carica elementare e; ma, come base degli esponenziali neperiani, sarebbe preferibile trattarla sempre come un operatore.
γ	$\gamma = 0{,}577\,215\ldots$	Vale la stessa osservazione fatta per π: γ.
i , j	unità immaginaria, operatore di rotazione	Va scritta in tondo come se fosse un operatore
...	omissione	Si usa sia nel significato di *elementi omessi* sia in quello di *eccetera*
x, y, z	coordinate cartesiane	x: larghezza, y: profondità, z: altezza
ϱ, φ, z	coordinate cilindriche	ϱ: distanza dall'asse, φ: longitudine, z: altezza
ϱ, φ, ϑ	coordinate sferiche	ϱ: distanza radiale, φ: longitudine, ϑ: colatitudine
$a = b$	uguale	
$a \neq b$	diverso	
$a \equiv b$	identico	
$\mathrm{e} \approx 2{,}718$	uguale a circa	
$a \sim b$	proporzionale	Si può usare anche $a \propto b$
$a \leftrightarrow b$	equivalente	
$a > b$	maggiore	
$a < b$	minore	
$a \geq b$	maggiore o uguale	
$a \leq b$	minore o uguale	
$a \gg b$	molto maggiore	
$a \ll b$	molto minore	

continua

Continua tabella 24.8

Simbolo	Significato	Note
$a \to b$	tendente	
$a \simeq b$	asintoticamente uguale	
$a \triangleq b$	corrispondente	Si usa nelle indicazioni di scala dei diagrammi: per esempio $1\,\text{cm} \triangleq 10\,\text{V}$
$\begin{cases}[a,b] \\ (a,b] \\ [a,b) \\ (a,b)\end{cases}$	intervallo	Intervallo "da a a b"; l'intervallo è chiuso all'estremo adiacente ad una parentesi quadra e aperto all'estremo adiacente ad una parentesi tonda.
$a + b$	somma	
$a - b$	sottrazione	
$ab\,,\ a \cdot b$	moltiplicazione	Non usare altri simboli quando gli operandi sono indicati mediante lettere
$1{,}5 \times 2{,}3$	moltiplicazione	Non usare altri simboli quando gli operandi sono entrambi numerici
$\begin{cases}1{,}5\,a \\ 1{,}5 \cdot a\end{cases}$	moltiplicazione	Gli operandi numerici precedono sempre quelli letterali
$a/b\,,\ \dfrac{a}{b}$	divisione	Le due simbologie possono essere mescolate; usare le parentesi per isolare le singole operazioni ed evitare ambiguità; per esempio $\dfrac{(a/b)+1}{(a/b)+(b/a)}$
$\operatorname{sgn} x$	segno	$\operatorname{sgn} x = \begin{cases}1 & \text{se } x > 0 \\ 0 & \text{se } x = 0 \\ -1 & \text{se } x < 0\end{cases}$
$\lfloor x \rfloor$	*floor*	Il massimo numero intero minore o uguale al numero reale x; esempi: $\lfloor 2{,}4 \rfloor = 2$; $\lfloor -2{,}4 \rfloor = -3$

continua

Continua tabella 24.8

Simbolo	Significato	Note
$\lceil x \rceil$	*ceiling*	Il minimo numero intero maggiore o uguale al numero reale x; esempi: $\lceil 2{,}4 \rceil = 3$; $\lceil -2{,}4 \rceil = -2$
$\operatorname{int} x$	parte intera	$\operatorname{int} x = \operatorname{sgn} x \cdot \lfloor \lvert x \rvert \rfloor$
$\operatorname{frac} x$	parte fratta	$\operatorname{frac} x = x - \operatorname{int} x$
$i \mod j$	modulo	Le norme ISO 80000-2:2009 2-18: definiscono 'mod' solo come operatore fra interi: se $i \in \mathbb{Z}$ e $j \in \mathbb{N}$ allora $m = i \mod j$ è è il resto della divisione intera di i/j
a^b	elevazione a potenza	
$\sqrt[b]{a}$	estrazione di radice	Non usare né $\sqrt[b]{}a$ né $\sqrt[b]{}(a)$; se $b = 2$, b viene omesso
$\lvert a \rvert$	valore assoluto	
$\sum_{i=1}^{n} a_i$	somma	
$\prod_{i=1}^{n} a_i$	prodotto	
$n!$	fattoriale	
$\binom{n}{m}$	coefficiente binomiale	$\dfrac{n(n-1)\cdots(n-m+1)}{1 \times 2 \times \cdots m}$
$f(x)$	funzione	
$\log_b x$	logaritmo in base b	
$\lg x$, $\log_{10} x$	logaritmo in base 10	
$\ln x$, $\log_{\mathrm{e}} x$	logaritmo naturale o neperiano	
$\operatorname{lb} x$, $\log_2 x$	logaritmo in base 2	

continua

Continua tabella 24.8

Simbolo	Significato	Note
e^x , $\exp x$	esponenziale	In questa e nelle funzioni successive scritte in caratteri tondi l'argomento non necessita di parentesi quando è composto da un solo elemento letterale o numerico
$\sin x$	seno	
$\cos x$	coseno	
$\tan x$	tangente	
$\cot x$	cotangente	
$\sinh x$	seno iperbolico	
$\cosh x$	coseno iperbolico	
$\tanh x$	tangente iperbolica	
$\coth x$	cotangente iperbolica	
$\arcsin x$	arcoseno	
$\arccos x$	arcocoseno	
$\arctan x$	arcotangente	
$\operatorname{arccot} x$	arcocotangente	
$\operatorname{arsinh} x$	arcoseno iperbolico	
$\operatorname{arcosh} x$	arcocoseno iperbolico	
$\operatorname{artanh} x$	arcotangente iperbolica	
$\operatorname{arcoth} x$	arcocotangente iperbolica	
$K(k)$	integrale ellittico completo di prima specie	$K(k) = \int_0^{\pi/2} \frac{\mathrm{d}\vartheta}{\sqrt{1 - k^2 \sin^2 \vartheta}}$
$F(\varphi, k)$	integrale ellittico incompleto di prima specie	$F(\varphi,\, k) = \int_0^{\varphi} \frac{\mathrm{d}\vartheta}{\sqrt{1 - k^2 \sin^2 \vartheta}}$

continua

Continua tabella 24.8

Simbolo	Significato	Note
$E(\varphi, k)$	integrale ellittico incompleto di seconda specie	$E(\varphi,k)=\int_0^{\varphi}\sqrt{1-k^2\sin^2\vartheta}\,\mathrm{d}\vartheta$
$\Pi(n;\varphi,k)$	integrale ellittico incompleto di terza specie	$\Pi(n;\varphi,k)=\int_0^{\varphi}\frac{\mathrm{d}\vartheta}{(1-n\sin^2\vartheta)\sqrt{1-k^2\sin^2\vartheta}}$
$\varphi(x,k)$	amplitudine	L'amplitudine è legata all'integrale ellittico di prima specie dalla relazione $x=F(\varphi,\,k)$
$\operatorname{sn}(x,k)$	seno ellittico	$\operatorname{sn}(x,k)=\sin\varphi$
$\operatorname{cn}(x,k)$	coseno ellittico	$\operatorname{cn}(x,k)=\cos\varphi$
$\begin{cases}\operatorname{dn}(x,\,k)\\ \Delta(\varphi)\end{cases}$	delta amplitudine	$\Delta(\varphi)=\sqrt{1-k^2\sin^2\varphi}$
$o(x)$	ordine di infinito o infinitesimo	Se $y=o(x)$ allora $\lim y/x=0$
$O(x)$	ordine di infinito o infinitesimo	Se $y=O(x)$ allora $\lvert\lim y/x\rvert<\infty$
$\Gamma(z)$	funzione gamma	$\Gamma(z)=\int_0^{\infty}t^{z-1}\,\mathrm{e}^{-t}\,\mathrm{d}t$
$(a)_n$	simbolo di Pochhammer	$(a)_n=\frac{\Gamma(a+n)}{\Gamma(a)}$
$\operatorname{erf}(z)$	funzione d'errore	$\operatorname{erf}(z)=\frac{2}{\sqrt{\pi}}\int_0^{z}\mathrm{e}^{-t^2}\,\mathrm{d}t$
$\operatorname{erfc}(z)$	funzione complementare d'errore	$\operatorname{erfc}(z)=1-\operatorname{erf}(z)$
$\mathrm{C}(z)$	integrale di Fresnel	$\mathrm{C}(z)=\int_0^{z}\cos(\pi t^2/2)\,\mathrm{d}t$
$\mathrm{S}(z)$	integrale di Fresnel	$\mathrm{S}(z)=\int_0^{z}\sin(\pi t^2/2)\,\mathrm{d}t$
$\mathrm{Si}(z)$	seno integrale	$\mathrm{Si}(z)=\int_0^{z}\frac{\sin t}{t}\,\mathrm{d}t$
$\mathrm{Ci}(z)$	coseno integrale	$\mathrm{Ci}(z)=\gamma+\ln z+\int_0^{z}\frac{\cos t-1}{t}\,\mathrm{d}t$

continua

Continua tabella 24.8

Simbolo	Significato	Note
$E_1(z)$	esponenziale integrale	$E_1(z) = \int_z^\infty \frac{\mathrm{e}^{-t}}{t}\,\mathrm{d}t$
$\mathrm{Ei}(x)$	esponenziale integrale	$\mathrm{Ei}(x) = \int_{-\infty}^x \frac{\mathrm{e}^{-t}}{t}\,\mathrm{d}t$
$\mathrm{li}(x)$	logaritmo integrale	$\mathrm{li}(x) = \int_0^x \frac{\mathrm{d}t}{\ln t} = \mathrm{Ei}(\ln x)$
$\zeta(s)$	funzione Zeta di Riemann	$\zeta(s) = \sum_{k=1}^\infty k^{-s}$
$\delta(t)$	distribuzione di Dirac	
$\mathrm{u}(t)$, $\mathrm{U}(t)$	gradino unitario	$\mathrm{u}(t) = \begin{cases} 0 & \text{per } t < 0 \\ 1/2 & \text{per } t = 0 \\ 1 & \text{per } t > 0 \end{cases}$
δ_{ij}	simbolo di Kronecker	$\delta_{ij} = \begin{cases} 0 & \text{per } i \neq j \\ 1 & \text{per } i = j \end{cases}$
$J_\nu(z)$	funzione di Bessel di prima specie	
$Y_\nu(z)$	funzione di Bessel di seconda specie	
$H_\nu^{(1)}(z)$	funzione di Hankel di prima specie	$H_\nu^{(1)}(z) = J_\nu(z) + \mathrm{i}\,Y_\nu(z)$
$H_\nu^{(2)}(z)$	funzione di Hankel di seconda specie	$H_\nu^{(2)}(z) = J_\nu(z) - \mathrm{i}\,Y_\nu(z)$
$I_\nu(z)$	funzione di Bessel modificata di prima specie	
$K_\nu(z)$	funzione di Bessel modificata di seconda specie	
$\mathrm{ber}_\nu(x)$	prima funzione di Kelvin di prima specie	$\mathrm{ber}_\nu(x) = \mathbf{Re}\left[J_\nu\left(x\,\mathrm{e}^{3\pi\,\mathrm{i}/4}\right)\right]$
$\mathrm{bei}_\nu(x)$	seconda funzione di Kelvin di prima specie	$\mathrm{bei}_\nu(x) = \mathbf{Im}\left[J_\nu\left(x\,\mathrm{e}^{3\pi\,\mathrm{i}/4}\right)\right]$
$\ker_\nu(x)$	prima funzione di Kelvin di seconda specie	$\ker_\nu(x) = \mathbf{Re}\left[K_\nu\left(x\,\mathrm{e}^{\pi\,\mathrm{i}/4}\right)\right]$

continua

Continua tabella 24.8

Simbolo	Significato	Note
$\mathrm{kei}_\nu(x)$	seconda funzione di Kelvin di seconda specie	$\mathrm{kei}_\nu(x) = \mathbf{Im}\left[K_\nu\left(x\,\mathrm{e}^{\pi\,\mathrm{i}\,/4}\right)\right]$
$M(a,\,b,\,z)$	funzione ipergeometrica confluente	Funzione di Kummer di prima specie
$U(a,\,b,\,z)$	funzione ipergeometrica confluente	Funzione di Kummer di seconda specie
$F(a,\,b;c;z)$	funzione ipergeometrica	L'espressione generale è $F(a,\,b;c;z) = \dfrac{\Gamma(c)}{\Gamma(b)\Gamma(c-b)} \cdot \displaystyle\int_0^1 t^{b-1}(1-t)^{c-b-1}(1-tz)^{-a}\,\mathrm{d}t$
$P_n(z)$	polinomio di Legendre	Detto anche funzione sferica
$T_n(z)$	polinomio di Chebyshev di prima specie	
$U_n(z)$	polinomio di Chebyshev di seconda specie	
$C_n^{(\alpha)}(z)$	polinomio di Gegenbauer	Detto anche polinomio ultrasferico
$P_n^{(\alpha,\,\beta)}(z)$	polinomio di Jacobi	L'intervallo di ortogonalità è $[-1,+1]$
$G(p,\,q,\,z)$	polinomio di Jacobi	L'intervallo di ortogonalità è $[0,+1]$
$H_n(z)$	polinomio di Hermite	
$L_n(z)$	polinomio di Laguerre	
$L_n^{(\alpha)}(z)$	polinomio di Laguerre generalizzato	
$B_n(z)$	polinomio di Bernoulli	
$E_n(z)$	polinomio di Eulero	
$P_\nu^\mu(z)$	funzione ultrasferica di prima specie	Quando $\mu = 0$ si omette di scriverne il valore, perché la funzione coincide con il polinomio di Legendre
$Q_\nu^\mu(z)$	funzione ultrasferica di seconda specie	
$\lim\limits_{x\to a} f(x)$	limite	

continua

Continua tabella 24.8

Simbolo	Significato	Note
Δx	incremento finito	
δx	incremento virtuale	
$\mathrm{d}x$	differenziale	Va scritto in tondo come se fosse un operatore, ma non richiede lo spazio a destra
$f(x)\Big\vert_a^b$	incremento	Cioè $f(b) - f(a)$
$\dfrac{\mathrm{d}y}{\mathrm{d}x}$	derivata	
$\dfrac{\partial y}{\partial x}$	derivata parziale	Font permettendo sarebbe desiderabile che il segno di derivata parziale fosse diritto, invece che inclinato; molti font matematici di tipo OpenType contengono questo simbolo diritto.
$\dfrac{\mathrm{d}^n y}{\mathrm{d}x^n}$	derivata n-esima	
$\dfrac{\partial^n y}{\partial x^n}$	derivata parziale n-esima	L'ordine di derivazione nelle derivate parziali miste, quando non sia indifferente, è il seguente $\dfrac{\partial^2 z}{\partial x \partial y} = \dfrac{\partial}{\partial x}\left(\dfrac{\partial z}{\partial y}\right)$
$\int f(x)\,\mathrm{d}x$	integrale indefinito	
$\int_a^b f(x)\,\mathrm{d}x$	integrale definito	
$\int_D f(\mathsf{P})\,\mathrm{d}D$	integrale esteso ad un dominio	Il particolare dominio D va specificato. Il punto P appartiene al dominio
$\fint_a^b f(z)\,\mathrm{d}z$	integrale principale di Cauchy	La funzione $f(z)$ è discontinua lungo la retta che congiunge a e b, e l'integrale viene calcolato come limite simmetrico a cavallo della discontinuità

continua

Continua tabella 24.8

Simbolo	Significato	Note
$\oint_\ell f(z)\,\mathrm{d}z$	circuitazione di f lungo la linea chiusa ℓ	
$\mathcal{L}[f(t)]$	trasformata di Laplace	$F(s) = \mathcal{L}[f(t)]$
$\mathcal{L}^{-1}[F(s)]$	antitrasformata di Laplace	$f(t) = \mathcal{L}^{-1}[F(s)]$
$\mathcal{F}[f(t)]$	trasformata di Fourier	$F(\omega) = \mathcal{F}[f(t)]$
$\mathcal{F}^{-1}[F(\omega)]$	antitrasformata di Fourier	$f(t) = \mathcal{F}^{-1}[F(\omega)]$
$\widehat{\mathsf{ABC}}$	angolo	Il vertice corrisponde al punto B
$\overset{\frown}{\mathsf{AB}}$	arco	
$\overline{\mathsf{AB}}$	segmento	
$\vec{V}$	vettore	
$\lvert\vec{V}\rvert\,,\, V$	modulo di vettore	
$\vec{V}_1 \cdot \vec{V}_2$	prodotto scalare	Non usare mai l'operatore $\times$ per il prodotto scalare
$\vec{V}_1 \times \vec{V}_2$	prodotto vettore	
$\int_\ell \vec{V} \cdot \overrightarrow{\mathrm{d}\ell}$	'lavoro' di $\vec{V}$ lungo la linea ℓ	
$\int_S \vec{V} \cdot \overrightarrow{\mathrm{d}S}$	flusso di $\vec{V}$ attraverso la superficie S	
$\operatorname{grad}\Phi\,,\, \nabla\Phi$	gradiente	Si può indicare anche con $\overrightarrow{\operatorname{grad}\Phi}$ oppure $\overrightarrow{\nabla\Phi}$
$\operatorname{div}\vec{V}\,,\, \nabla\cdot\vec{V}$	divergenza	
$\operatorname{rot}\vec{V},\ \nabla\times\vec{V}$	rotore	Si può indicare anche con $\overrightarrow{\operatorname{rot}\vec{V}}$ oppure $\overrightarrow{\nabla\times\vec{V}}$
$\nabla^2\Phi$	laplaciano di uno scalare	

continua

Continua tabella 24.8

Simbolo	Significato	Note
$\nabla^2\vec{V}$	laplaciano di un vettore	Vettore le cui componenti sono ordinatamente i laplaciani delle componenti di $\vec{V}$. Si può indicare anche con $\overrightarrow{\nabla^2\vec{V}}$
x	valore istantaneo	
X	valore efficace	Il concetto ha senso solo se $x(t)$ è periodica
$\hat{x}$, $x_{\max}$	valore massimo	Il valore massimo della funzione $x(t)$ in un intervallo prefissato $[t_{\min}, t_{\max}]$
$\check{x}$, $x_{\min}$	valore minimo	Il valore minimo della funzione $x(t)$ in un intervallo prefissato $[t_{\min}, t_{\max}]$
$\bar{x}$	valore medio	Il valore medio della funzione $x(t)$ in un intervallo prefissato $[t_{\min}, t_{\max}]$
$\mathbf{Re}\, z$	parte reale	
$\mathbf{Im}\, z$	parte immaginaria	Se $z = x + \mathrm{i}\, y$ allora $\mathbf{Im}\, z$ è uguale a y e non a $\mathrm{i}\, y$, che è la componente immaginaria di z
$\lvert z\rvert$	modulo	
$\arg z$	argomento o anomalia	$z = \lvert z\rvert\, \mathrm{e}^{\mathrm{i}\arg z}$
z^*	coniugato	Nei testi matematici è più comune $\bar{z}$
$f_*(s)$	paraconiugato	$f_*(s) = f(-s)$, ma se $f(s)$ è hermitiana, $f_*(\mathrm{i}\, y) = f^*(\mathrm{i}\, y)$
$\mathbf{A}$	insieme	$\mathbf{A} = \{a_1, a_2, \dots\}$. Nello stesso modo, per indicare altri insiemi, si possono usare altre lettere maiuscole, che non siano già associate ad insiemi particolari
$\emptyset$	insieme vuoto	
Ω	universo	

continua

Continua tabella 24.8

Simbolo	Significato	Note
$\mathbf{N}$, $\mathbb{N}$	insieme dei numeri interi positivi	
$\mathbf{Z}$, $\mathbb{Z}$	insieme dei numeri interi relativi	
$\mathbf{Q}$, $\mathbb{Q}$	insieme dei numeri razionali	
$\mathbf{R}$, $\mathbb{R}$	insieme dei numeri reali	
$\mathbf{C}$, $\mathbb{C}$	insieme dei numeri complessi	
$\mathbf{A} \times \mathbf{B}$	prodotto cartesiano di insiemi	Ogni elemento del prodotto cartesiano è formato dall'accoppiamento di un elemento dell'insieme $\mathbf{A}$ con un elemento dell'insieme $\mathbf{B}$
$\mathbf{R}^n$, $\mathbb{R}^n$	insieme delle n-uple reali	Indica anche lo spazio reale a n dimensioni
$\mathbf{C}^n$, $\mathbb{C}^n$	insieme delle n-uple complesse	Indica anche lo spazio complesso ad n dimensioni
$\exists$	esiste	Esiste ed è unico: $\exists!$
$\nexists$	non esiste	
$a \in \mathbf{A}$	appartiene	
$a \notin \mathbf{A}$	non appartiene	
$\mathbf{A} \ni a$	contiene	
$\mathbf{A} \not\ni a$	non contiene	
$\mathbf{A} \cap \mathbf{B}$	intersezione	
$\mathbf{A} \cup \mathbf{B}$	unione	
$\mathbf{A} \backslash \mathbf{B}$	differenza	L'insieme $\mathbf{A} \backslash \mathbf{B}$ è formato dagli elementi di $\mathbf{A}$ esclusi quelli che appartengono anche a $\mathbf{B}$
$\complement_{\Omega}\mathbf{A}$	complemento	$\complement_{\Omega}\mathbf{A} = \Omega \backslash \mathbf{A}$
$\mathbf{A} \subset \mathbf{B}$	è contenuto	$\mathbf{A}$ è un sottoinsieme di $\mathbf{B}$

continua

Continua tabella 24.8

Simbolo	Significato	Note
$\mathbf{A} \not\subset \mathbf{B}$	non è contenuto	
$\mathbf{A} \subseteq \mathbf{B}$	è contenuto o coincide	
$\mathbf{A} \nsubseteq \mathbf{B}$	non è contenuto né coincide	
$\mathbf{B} \supset \mathbf{A}$	contiene	**B** contiene l'insieme **A**
$\mathbf{B} \not\supset \mathbf{A}$	non contiene	
$\mathbf{B} \supseteq \mathbf{A}$	contiene o coincide	
$\mathbf{B} \not\supseteq \mathbf{A}$	non contiene né coincide	
$\boldsymbol{A}$	matrice	Quando la matrice ha una sola colonna (riga) si è soliti usare lettere minuscole. La matrice può essere esplicitata in uno dei modi seguenti $\boldsymbol{A} = \begin{pmatrix} a_{11} & \dots & a_{1r} \\ \vdots & \ddots & \vdots \\ a_{n1} & \dots & a_{nr} \end{pmatrix}$ oppure $\boldsymbol{A} = \begin{bmatrix} a_{11} & \dots & a_{1r} \\ \vdots & \ddots & \vdots \\ a_{n1} & \dots & a_{nr} \end{bmatrix}$
$\lvert\boldsymbol{A}\rvert$, $\det \boldsymbol{A}$	determinante	La matrice $\boldsymbol{A}$ di cui si calcola il determinante deve essere quadrata. È $\det \boldsymbol{A} = \begin{vmatrix} a_{11} & \dots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \dots & a_{nn} \end{vmatrix}$
$\lVert\boldsymbol{A}\rVert$	norma	
$\operatorname{tr} \boldsymbol{A}$	traccia	$\operatorname{tr} \boldsymbol{A} = \sum_{i=1}^{n} a_{ii}$
$\boldsymbol{A}^*$	matrice coniugata	Nei testi di matematica è più frequente $\bar{\boldsymbol{A}}$
$\tilde{\boldsymbol{A}}$, ${}^t\boldsymbol{A}$	matrice trasposta	Si indica anche con $\boldsymbol{A}^{\mathrm{T}}$. Questo secondo simbolo è più adatto quando la matrice è esplicitata mediante i suoi elementi
$\bar{\boldsymbol{A}}$	matrice associata	$\bar{\boldsymbol{A}} = \tilde{\boldsymbol{A}}^*$

continua

Continua tabella 24.8

Simbolo	Significato	Note
$\boldsymbol{A}_*$	matrice paraconiugata	$\boldsymbol{A}_*(s) = \tilde{\boldsymbol{A}}(-s)$, ma se gli elementi $a_{ij}(s)$ di $\boldsymbol{A}$ sono hermitiani $\forall\, i, j$, è $\boldsymbol{A}_*(\mathrm{i}\, y) = \bar{\boldsymbol{A}}(\mathrm{i}\, y)$
$\boldsymbol{D}$	matrice diagonale	$\boldsymbol{D} = \boldsymbol{diag}\,(a_1, \ldots, a_n)$. Può essere usata qualunque altra lettera, purché ne sia definito il significato
$\boldsymbol{I}$, $\boldsymbol{1}$	matrice identità	Il secondo simbolo può essere usato solo quando il primo possa ingenerare confusione con altre grandezze
$\boldsymbol{\mathcal{G}}$	diadica	
$s_{\overline{n}\rceil i}$	capitalizzazione	Coefficiente di capitalizzazione di n annualità posticipate all'interesse i
$\ddot{s}_{\overline{n}\rceil i}$	capitalizzazione	Coefficiente di capitalizzazione di n annualità anticipate all'interesse i
$a_{\overline{n}\rceil i}$	attualizzazione	Coefficiente di attualizzazione di n annualità posticipate all'interesse i
$\ddot{a}_{\overline{n}\rceil i}$	attualizzazione	Coefficiente di attualizzazione di n annualità anticipate all'interesse i
$\sigma_{\overline{n}\rceil i}$	reintegrazione	Coefficiente di reintegrazione mediante n annualità posticipate all'interesse i
$\ddot{\sigma}_{\overline{n}\rceil i}$	reintegrazione	Coefficiente di reintegrazione mediante n annualità anticipate all'interesse i
$\alpha_{\overline{n}\rceil i}$	ammortamento	Coefficiente di ammortamento mediante n annualità posticipate all'interesse i
$\ddot{\alpha}_{\overline{n}\rceil i}$	ammortamento	Coefficiente di ammortamento mediante n annualità anticipate all'interesse i

Tabella 24.9: Nomenclatura, simboli e unità di misura

Grandezza	**Simbolo**	**Unità SI**
angolo piano	α, β, γ, ...	rad
angolo solido	ω, Ω	sr
lunghezza	l	m
larghezza	b	m
altezza	h	m
raggio	r	m
spessore	d, δ	m
diametro	d	m
percorso curvilineo	s	m
superficie, area	S, A	m^2
volume	V, v	m^3
lunghezza d'onda	λ	m, (m/onda)
numero d'onda ($1/\lambda$)	σ	m^{-1}, (onde/m)
ondulanza ($2\pi/\lambda$)	k	m^{-1}
attenuazione spaziale	α	m^{-1}, (Np/m)
costante di fase	β	m^{-1}
costante di propagazione ($\alpha + \mathrm{i}\,\beta$)	γ	m^{-1}
tempo	t	s
periodo	T	s, (s/ciclo)
frequenza	f	Hz, (cicli/s)
pulsazione	ω	s^{-1}
tempo di rilassamento o costante di tempo	τ	s, (s/Np)
coefficiente di smorzamento	δ	s^{-1}, (Np/s)
decremento logaritmico (T/τ)	Λ	(Np/ciclo)
velocità	v, u	m/s

continua

Continua tabella 24.9

Grandezza	**Simbolo**	**Unità SI**
velocità angolare	ω	rad/s
accelerazione	a	$\mathrm{m/s^2}$
accelerazione angolare	α	$\mathrm{rad/s^2}$
accelerazione di gravità	g	$\mathrm{m/s^2}$
costante di gravitazione	G	$\mathrm{N\,m^2/kg^2}$
velocità della luce nel vuoto	c_0	m/s
massa	m	kg
massa volumica	ϱ	$\mathrm{kg/m^3}$
densità relativa (all'acqua)	d	–
volume massico ($1/\varrho$)	v	$\mathrm{m^3/kg}$
quantità di moto	p	kg m/s
momento della quantità di moto	L	$\mathrm{kg\,m^2/s}$
momento quadratico di superficie	I	$\mathrm{m^4}$
momento di inerzia	J	$\mathrm{kg\,m^2}$
forza	F	N
coppia	T, M	N m, (N m/rad)
momento di una forza	M	N m, (N m/rad)
pressione	p	Pa
tensione normale	σ	Pa
tensione di taglio	τ	Pa
allungamento relativo	ε	–
modulo di elasticità	E	Pa
angolo di torsione	γ	rad
modulo di torsione	G	Pa
dilatazione volumica relativa	ϑ	–

continua

Continua tabella 24.9

Grandezza	Simbolo	Unità SI
modulo di compressione	K	Pa
rapporto di Poisson	μ	–
viscosità dinamica	η	Pa s
viscosità cinematica (η/ϱ)	ν	m^2/s
coefficiente di attrito	μ	–
tensione superficiale	γ, σ	N/m
energia	E	J
energia potenziale	E_p, V, Φ	J
energia cinetica	E_k, T, K	J
lavoro	W	J
potenza	P	W
rendimento	η	–
velocità del suono	c	m/s
velocità longitudinale	c_l	m/s
velocità trasversale	c_t	m/s
velocità di gruppo	c_g	m/s
flusso energetico (acustico)	P	W/m^2
fattore di riflessione (acustica)	ϱ	–
fattore di assorbimento (acustico)	α_a, α	–
fattore di trasmissione (acustica)	τ	–
fattore di dissipazione (acustica)	δ	–
livello sonoro	L_N, Λ	dB
corrente elettrica	i, I	A
densità di corrente	j, J	A/m^2
carica elettrica	Q	C

continua

Continua tabella 24.9

Grandezza	**Simbolo**	**Unità SI**
densità volumica di carica	ϱ	$\mathrm{C/m^3}$
densità superficiale di carica	σ	$\mathrm{C/m^2}$
potenziale elettrico	V	V
tensione (elettrica)	V	V
impulso di tensione	U	V s
forza elettromotrice	E	V
campo elettrico	E, K	V/m
spostamento elettrico	D	$\mathrm{C/m^2}$
flusso elettrico	Ψ	C
capacità	C	F
permittività (o permettività)	ε	F/m
permittività del vuoto	ε_0	F/m
permittività relativa	ε_r	–
polarizzazione dielettrica	P	$\mathrm{C/m^2}$
suscettività elettrica ($\varepsilon_\mathrm{r} - 1$)	χ_e	–
elettrizzazione ($D/\varepsilon_0 - E$)	E_i, K_i	V/m
polarizzazione ($D - \varepsilon_0 E$)	P	$\mathrm{C/m^2}$
momento di dipolo (elettrico)	p	C m
campo magnetico	H	A/m
potenziale magnetico	U_m	A
forza magnetomotrice	F_m	A
induzione magnetica	B	T
flusso di induzione (magnetica)	Φ	Wb
permeabilità	μ	H/m
permeabilità del vuoto	μ_0	H/m

continua

Continua tabella 24.9

Grandezza	Simbolo	Unità SI
permeabilità relativa	μ_r	–
magnetizzazione	M	A/m
suscettività magnetica ($\mu_\mathrm{r} - 1$)	χ_m, κ	–
momento elettromagnetico	m, μ	$\mathrm{A\,m^2}$
polarizzazione magnetica	J	T
resistenza	R	Ω
reattanza	X	Ω
impedenza	Z	Ω
fattore di qualità	Q_L, Q_C, ...	–
coefficiente di risonanza	Q	–
conduttanza	G	S
suscettanza	B	S
ammettenza	Y	S
resistività	ϱ	$\Omega\,\mathrm{m}$
conducibilità	σ, γ	S/m
induttanza (propria)	L	H
induttanza mutua	M	H
coefficiente di accoppiamento ($M/\sqrt{L_\mathrm{p} L_\mathrm{s}}$)	k	–
coefficiente di dispersione ($1 - k^2$)	σ	–
riluttanza	$\mathcal{R}$, R	$\mathrm{H^{-1}}$
permeanza	Λ	H
potenza reattiva	Q	V A
potenza apparente	P	V A
sfasamento	φ	rad
numero delle fasi	m	–

continua

Continua tabella 24.9

Grandezza	**Simbolo**	**Unità SI**
angolo di perdita	δ	rad
numero di spire	N, n	–
densità volumica di energia elettromagnetica	w	$\mathrm{J/m^3}$
vettore di Poynting	S	$\mathrm{W/m^2}$
potenziale vettore magnetico	$\mathcal{A}$, A	Wb/m
temperatura termodinamica	T	K
temperatura (Celsius)	t	°C
quantità di calore	Q	J
entropia	S	J/K
energia interna	U	J
energia libera ($U - TS$)	F	J
entalpia	H	J
entalpia libera	G	J
coefficiente di pressione $(\partial(\ln p)/\partial T\|_V)$	β	$\mathrm{K^{-1}}$
compressibilità $(-\partial(\ln V)/\partial p\|_T)$	κ	$\mathrm{m^2/N}$
coefficiente di dilatazione lineare	α	$\mathrm{K^{-1}}$
coefficiente di dilatazione volumica	γ	$\mathrm{K^{-1}}$
conducibilità termica	λ	$\mathrm{W/(m\,K)}$
calore massico	c_p, c_v	$\mathrm{J/(kg\,K)}$
capacità termica	C_p, C_v	J/K
rapporto dei calori massici	κ	–
flusso termico	Φ	W
flusso di calore areico	q	$\mathrm{W/m^2}$
coefficiente di trasmissione termica	τ	$\mathrm{W/(m^2K)}$
coefficiente di diffusione termica	a	$\mathrm{m^2/s}$

continua

Continua tabella 24.9

Grandezza	Simbolo	Unità SI
potenza raggiante	Q, W	W
intensità energetica	I	W/sr
irradiamento	E	$\mathrm{W/m^2}$
radianza	L	$\mathrm{W/(m^2\,sr)}$
intensità luminosa	I	cd
flusso luminoso	Φ	lm
quantità di luce	Q	lm s
luminanza	L	$\mathrm{cd/m^2}$
illuminamento	E	lx
fattore di assorbimento	α	–
fattore di riflessione	ϱ	–
fattore di trasmissione	τ	–
indice di rifrazione	n	–
distanza di due piani reticolari adiacenti	d	m
angolo di Bragg	ϑ	rad
massa efficace dell'elettrone	m^*, m_{eff}	kg
energia di Fermi	E_{F}	J
vettore d'onda di Fermi	k_{F}	$\mathrm{m^{-1}}$, (rad/m)
coefficiente di Seebeck	S	V/K
coefficiente di Peltier	Π	V
coefficiente piezoelettrico (polarizzazione/sforzo)	d_{mn}	C/N
temperatura caratteristica di Weiss	Θ_{W}	K
temperatura di Curie	T_{C}	K
temperatura di Neel	T_{N}	K
coefficiente di Hall	R_{H}	$\mathrm{V\,m^2/A^2}$

continua

Continua tabella 24.9

Grandezza	**Simbolo**	**Unità SI**
quantità di sostanza	n	mol
massa molare	M	kg/mol
volume molare	V_m	m^3/mol
energia interna molare	U_m	J/mol
capacità termica molare	$C_{p\mathrm{m}}$, $C_{v\mathrm{m}}$	J/(mol K)
entropia molare	S_m	J/(mol K)
concentrazione di un costituente	c	$\mathrm{mol/m}^3$
molalità di un soluto	m	mol/kg
dose assorbita	D	Gy
energia impartita massica	z	Gy
indice di dose assorbita	D_i	Gy
rateo di dose assorbita	$\dot{D}$	Gy/s
equivalente di dose	H	Sv
esposizione	X	C/kg
rateo di esposizione	$\dot{X}$	C/(kg s)
attività di un radio nuclide	A	Bq

Nota bene I simboli delle unità di misura si usano solo se preceduti dal valore numerico della misura; non si usano mai nelle formule dove si indicano le grandezze con i loro simboli. Nell'indicazione delle scale dei diagrammi le unità di misura non si indicano mai fra parentesi quadre, al massimo fra parentesi tonde. Lo stesso vale per le intestazioni delle colonne delle tabelle. In metrologia le parentesi quadre hanno un significato particolare che ne esclude l'uso per specificare le unità di misura; anche espressioni verbali o testuali come "... dopo t secondi..." sono errate; invece è corretto dire o scrivere "... dopo un tempo t...".

Capitolo 25

Divisione in sillabe

Conviene specificare subito una differenza fra le due parole *sillabazione* e *cesura*. La sillabazione è il metodo che segue le regole grammaticali di una data lingua per dividerne le parole in sillabe. La cesura (in fin di riga) è il metodo che serve al tipografo per dividere una parola in fin di riga in modo da ottemperare anche alle regole tipografiche, oltre che grammaticali.

In tipografia le regole servono, come tutte le altre di quest'arte, per rendere la lettura più agevole al lettore. Fra le regole tipografiche, per esempio c'è quella di non dividere nessuna parola in posizioni che lascino un frammento troppo corto prima e/o dopo la cesura; per esempio, in italiano la parola 'idea' secondo la grammatica di divide in sillabe come *i-de-a*; non ci sono altri punti di possibile divisione; la tipografia non la divide affatto perché la prima e l'ultima sillaba sono sono entrambe troppo corte; ma se si prende la stessa parola preceduta da una preposizione articolata, come 'dell'idea', dove l'apostrofo, sostituendo la vocale elisa, la sostituisce a tutti gli effetti, la divisione grammaticale sarebbe *del-l'i-de-a*, ma i punti di cesura tipografica sono solo *del-l'i-dea*, perché ora solo l'ultimo frammento è troppo corto. I programmi del sistema TEX sono in grado di eseguire la cesura per tutte le lingue che è in grado di gestire; qualche rarissima volta sbaglia (specialmente in inglese), ma dispone dei mezzi per correggere questi insoliti errori. Alcuni word processor, invece, sanno gestire l'inglese, ma, per esempio, in italiano non riconoscono la funzione vocalica dell'apostrofo usato per l'elisione.

L'algoritmo per dividere le parole in fin di riga è una specialità del sistema TEX, ed è stata incorporata in diversi altri programmi, tra i quali merita segnalare OpenOffice e LibreOffice come esempi di software Open Source, a cui possono collaborare tutti gli utenti che ne hanno la possibilità e la competenza; un po' come succede per il sistema TEX.

Oltre a quei due word processor, usano questo stesso metodo anche Apache OpenOffice, Inkskape, GIMP, Scribus, InDesign, Illustrator. Lo usano anche il client JavaScript con il modulo `hyphenator.js`, e i browser Web e i client di posta elettronica Gecko di Mozilla (Firefox per calcolatori normali e per

dispositivi mobili); WebKit di Apple e Linux (Safari, Konqueror); Blink di Google (Chromium e Chrome, Opera, Web Browser per Android). La lettura di questo capitolo permette quindi di capire il comportamento di molti programmi di uso comune, che non hanno legami con il sistema TEX, ma che hanno importato da questo il suo algoritmo di cesura.

25.1 Quando viene eseguita la cesura

TEX attiva l'algoritmo di sillabazione quando, terminata la lettura di un capoverso dal file sorgente e, dopo aver trasformato le informazioni contenute nel file sorgente in una lunga linea di caratteri (oltre altre informazioni specifiche) con l'aiuto dei pacchetti *inputenc* e *fontenc*, deve accingersi a spezzare questa lunga riga nelle varie righe giustificate che appariranno nel file di uscita.

Si noti bene: tutte le macro presenti nel file sorgente sono state già elaborate al punto che la lunga riga contiene solo caratteri e alcuni comandi 'primitivi' che non producono testo e che il motore di composizione usa solo nel momento di creare la pagina di uscita.

Per quel che riguarda la sillabazione si noti che il file di ingresso poteva contenere caratteri codificati in diversi modi, mentre il file di uscita, e quindi la lunga riga, contiene solo caratteri codificati secondo le prescrizioni dei font usati per l'uscita. Se si usa il vecchio encoding OT1 per i font di uscita, qualunque lettera con segni diacritici non è compresa nella polizza dei caratteri e deve venire ottenuta componendo il carattere di base con il segno diacritico che gli compete; questa composizione si ottiene per sovrapposizione; il comando per eseguire questa sovrapposizione è quindi un comando primitivo ancora presente nella stringa che costituisce la lunga riga da spezzare. Se invece si usa la codifica T1, le lettere con diacritici delle lingue occidentali sono già tutte presenti nella polizza del font usato e quindi la lunga riga contiene solo lettere (sempre trascurando le altre informazioni codificate che verranno usate solo nel processo di formazione della pagina di uscita, alcune delle quali solo durante la scrittura nel file di uscita di una pagina completa).

Ciò premesso, bisogna capire come fa il programma di composizione a individuare una parola, ciò che il programma crede che sia una parola; per quel programma una parola è una sequenza di lettere maiuscole o minuscole, tratte dallo stesso font, che comincia dopo uno spazio o dopo segni di interpunzione in senso lato (per esempio: parentesi, virgolette, ecc.) e finisce con il primo token che non abbia le caratteristiche di una lettera. Questa stringa non può essere seguita da token diversi da una lettera che appartengano a certe categorie, per esempio non può essere seguita dal comando `\footnote`. Non può nemmeno essere preceduta da una macro, perché lo spazio che separa la macro dalla stringa di lettere serve già all'interprete per capire che la macro è terminata; la stringa può, però, costituire l'argomento di una macro. Non deve nemmeno essere la prima parola dopo un cambiamento di modo, per esempio da verticale

a orizzontale; questo significa che la prima parola di un capoverso non viene mai divisa in sillabe.

Lavorando con pdflatex il lettore può sfruttare il comando `\showhyphens`[1] per vedere come il programma divida in sillabe; usando *babel* con l'opzione per l'italiano potrebbe vedere che cosa succede scrivendo;

```
\showhyphens{macroistruzione macro"istruzione}
```

e troverebbe sullo schermo o nel file `.log` un messaggio del tipo

```
Underfull \hbox (badness 10000) in paragraph at lines 5300--5300
[] \T1/lmr/m/n/10 ma-croi-stru-zio-ne ma-cro-istru-zio-ne
```

A seconda del sistema operativo e della particolare distribuzione del sistema TEX, il nome `\T1/lmr/m/n/10` potrebbe essere sostituito da qualcosa d'altro; in ogni caso esso rappresenta la descrizione interna del font in uso nel momento in cui viene eseguito il comando `\showhyphens`.

L'annotazione 'Underfull \hbox (badness 10000) in paragraph at lines ...' può venire ignorata; questo è il messaggio che il programma emette ogni volta che deve allargare troppo lo spazio interparola per giustificare una riga.

La parte più interessante viene dopo; infatti 'ma-croi-stru-zio-ne' è la divisione in sillabe che il programma eseguirebbe spontaneamente, senza sapere che si tratta di una parola composta, mentre 'ma-cro-istru-zio-ne' è la divisione in sillabe conseguente all'uso del carattere attivo `"`, che, come si è già detto, serve appunto (tra l'altro) per indicare una divisione etimologica, piuttosto che fonetica.

Si noti che la potenziale parola talvolta non coincide con quello che gli umani sanno essere una parola della loro lingua o di un'altra lingua straniera.

Se si usasse la codifica OT1, invece della codifica T1, gli accenti vengono collocati sulle vocali (o sopra o sotto altre lettere, come per esempio: ç oppure č in molte altre lingue diverse dall'italiano) mediante un comando primitivo che si chiama `\accent`; quindi anche se uno scrive `qualità`, quando il programma interpreta il contenuto del file sorgente, sostituisce la 'à' con una sequenza di istruzioni, in particolare sostituisce la stringa iniziale con `qualit\accent18a`; in questo modo per TEX quello che lui crede essere una parola è la stringa 'qualit'; all'interno di questa stringa esso trova un solo punto di divisione, 'qua-lit' e quindi l'effetto complessivo sarebbe di dividere la parola in 'qua-lità' invece che in 'qua-li-tà'.

Queste cesure mancate potrebbero essere talvolta causa di composizioni in cui qualche riga sporge fuori dalla giustezza, oppure qualche riga è stata spaziata troppo per consentire la giustificazione; tuttavia il problema è facilmente risolubile; basta usare nel preambolo la direttiva

[1] Il comando `\showhyphens` va usato con cautela con i programmi di composizione xelatex e lualatex. Qualunque sia il programma di composizione il lettore può caricare il pacchetto *testhyphens* e controllare la sillabazione delle parole inserendole dentro l'ambiente *checkhyphens*; le parole divise in sillabe non appaiono nel file `.log` ma direttamente nel documento.

```
\usepackage[T1]{fontenc}
```

come si è già detto diverse volte.

Quando si esamina il file `.log` per cercare di capire perché la mancata cesura in fin di riga abbia provocato righe troppo lunghe o spaziate, si scopre sempre che le cause sono quelle descritte sopra: una macro precede troppo da vicino o segue troppo da vicino una stringa di lettere che potrebbe essere una parola della lingua; oppure, bisogna sottolinearlo, si è usato un font che non consente la cesura, come per esempio un font della famiglia a spaziatura fissa.

Un espediente elementare sarebbe quello di dividere a mano: basta inserire il comando `\-` all'interno di una parola nel punto in cui si vorrebbe eseguire la cesura; in questo testo non è mai stato usato.

Ma più in generale basta inserire una spaziatura orizzontale di ampiezza nulla fra la parola e l'oggetto 'troppo vicino', come per esempio una macro, o una nota, o all'inizio di un capoverso, o prima della prima parola che segue un comando `\item` in una lista. Ciò si ottiene usando comandi di basso livello come per esempio

```
\hskip 0pt
```

dove `\hskip` è il comando primitivo che inserisce lo spazio orizzontale specificato immediatamente dopo, nel nostro caso 0 pt.

In certi casi questa operazione diventa risolutiva; faccio solo un esempio: la macro troppo vicina potrebbe essere una dichiarazione di cambiamento di font; il meccanismo di sillabazione non funziona, quindi, per la prima parola dopo un cambiamento di font; raramente questo è un vero problema, ma si presenta più frequentemente di quanto non si creda. Un comando che implica un cambiamento di font è `\emph` che cambia il font corrente in uno che non abbia la stessa inclinazione, cioè da tondo passa a corsivo e viceversa; da inclinato passa a tondo, e via di questo passo. Se si dovesse scrivere "nell'*eventualità*", nel file sorgente si inserisce `nell'\emph{eventualità}`. Fin qui è scontato e tutto dovrebbe svolgersi normalmente; ma se questa breve locuzione capita verso la fine della riga, non può venire divisa in sillabe, come si vede bene dalla *minipage* riquadrata di sinistra:

Bisogna osservare che nell'*eventualità*

Bisogna osservare che nell'*eventualità*

Si osserva infatti che l'ultima locuzione non è divisa in sillabe e che per mantenere la composizione giustificata il programma di composizione ha inserito spazi inaccettabili nelle righe precedenti. Però nella *minibox* riquadrata di destra è stato aggiunta la specificazione `\hskip 0pt` prima della parola 'eventualità', quando a rigore il cambiamento di font avviene prima che il programma prenda in considerazione quella parola e quindi la sillabazione avviene correttamente. Nel primo caso invece il programma considerava come "parola" tutta la stringa

`nell'\emph{eventualità}` all'interno della quale avveniva un cambiamento di font.

Ora il comando `\emph` richiama al suo interno la dichiarazione `\em` che presiede al cambiamento di font; se noi modifichiamo questa macro aggiungendole in coda il grumo di colla di larghezza nulla che separa quindi la dichiarazione dalla parola da sillabare, potremmo risolvere il problema; infatti usiamo uno dei comandi del pacchetto *etoolbox* già descritti in parte nel paragrafo 29.8.4.2. Usiamo il comando (robusto) `\addto` che permette di aggiungere in coda altro codice alla definizione di una macro; la sintassi è la seguente"

`\addto{`⟨*macro*⟩`}{`⟨*codice aggiuntivo*⟩`}{`⟨*vero*⟩`}{`⟨*falso*⟩`}`

Noi pertanto nel preambolo o in un file di macro personali scriviamo:

```
\addto{\em}{\nobreak\hskip0pt\relax}{}{}
```

dove il comando `\nobreak`, quando si parla di comandi che implicano un cambiamento di font, è quanto mai opportuno se prima del comando, come nel nostro caso, non c'è nessuno spazio, anzi è vietato andare a capo dopo l'apostrofo; dopodiché l'esempio precedente diventa:

Bisogna osservare che nell'*eventualità*

Come si vede ora la sillabazione funziona correttamente. Definendo un breve comando come `\hz`:

```
\newcommand*\hz{\nobreak\hskip0pt\relax}
```

in un file di macro personali o nel preambolo, si può anche inserire a mano questo breve comando in ogni posizione in cui sia necessario un grumetto di gomma di larghezza nulla per separare ciò che precede troppo da vicino una parola impedendone la sillabazione. In modo analogo si può operare (ma solo scrivendo in italiano) mediante l'uso del carattere attivo `"`, come descritto nel prossimo paragrafo.

È bene conoscere questo meccanismo al fine di porre rimedio alle rare circostanze in cui il meccanismo di sillabazione non svolga il suo compito come ci si aspetterebbe.

25.2 La sillabazione fonetica oppure etimologica

Vale la pena di aggiungere una osservazione. Nessun meccanismo automatico è infallibile; la sillabazione in italiano funziona molto bene, ma non si può escludere che un neologismo particolare o una parola tecnica possa risultare divisa in sillabe in modo scorretto, specialmente se è ricavato da una radice

straniera. Per esempio la parola tecnica dell'ambito medico 'dispepsia" viene divisa in 'di-spep-sia'; ovviamente questo è un caso semplice da trattare mediante l'uso del carattere attivo `"`, perché scrivendo `dis"pepsia`, la parola risulta divisa in modo etimologico. Dalla chimica si potrebbe prendere un lunga parola composta, per esempio *desclorfeniraminacloridrato* che può venire scritta nella forma `des"clor"fenir"amina"clor"idrato` che porta sempre alla divisione in sillabe corretta, anche nei punti non esplicitamente indicati: *des-clor-fe-nir-ami-na-clor-idra-to*. Attenzione: perché il moncone di parola *-ami-* non viene diviso in *-a-mi-*? Perché il segno `"` spezza la stringa che il programma crede essere una parola in tanti monconi che il programma tratta come parole intere, e in nessuna parola il programma stacca una sillaba iniziale o finale formata da una sola lettera; questo fatto verrà commentato meglio più avanti.

In inglese il meccanismo di sillabazione sembra funzionare correttamente in poco più del 90% dei casi. In francese l'algoritmo funzionerebbe benissimo, nonostante le numerose lettere con diacritici, ma le regole della buona educazione impediscono di dividere in sillabe parole che lascino nella riga contenente la cesura, o nella riga seguente, monconi di parole che hanno significati volgari; a causa di ciò alcune parole non vengono divise affatto o vengono divise in monconi talvolta troppo lunghi. In tedesco il problema è veramente difficile a causa delle numerosissime parole composte che vanno divise etimologicamente, talvolta con un cambio di ortografia. Tuttavia l'algoritmo viene usato con soddisfazione da tutti, e nei pochi casi in cui fallisce ci sono almeno tre strade assai semplici da seguire:

1. Si marcano a mano con `\-` i punti di divisione di quelle poche parole che nelle bozze risultano divise in modo scorretto; con l'italiano ciò è estremamente raro, per non dire impossibile, perché le norme UNI che regolano la materia ammettono come corrette tutte le divisioni fonetiche; certo con parole straniere o derivate da radici straniere questo potrebbe non essere corretto.

2. Scrivendo in italiano si usa il carattere attivo `"` nelle parole composte che si desidera dividere etimologicamente; si noti che le norme UNI che regolano questa questione in italiano accettano la divisione fonetica anche per le parole composte, ma in uno scritto tecnico è preferibile usare la divisione etimologica. Nelle altre lingue straniere, tranne in inglese, le opzioni di *babel* prevedono quasi sempre un comando che utilizza il carattere attivo `"`, per esempio `"-`, che consente di svolgere la stessa funzione che `"` da solo svolge in italiano; bisogna consultare la parte di documentazione di *babel* relativa alla lingua specifica. Dalla versione 3.9 di *babel* la documentazione relative alle lingue si legge semplicemente dando da terminale il comando `texdoc babel-`⟨*lingua*⟩ (per esempio `texdoc babel-italian` oppure `texdoc babel-german`, eccetera).

3. Si ricorre alla lista di parole divise in sillabe contenute nell'argomento del comando `\hyphenation`; per l'italiano, per esempio, se si dovesse usare

spesso la parola 'macroistruzione' al singolare e al plurale basterebbe scrivere nel preambolo:

```
\hyphenation{ma-cro-istru-zio-ne ma-cro-istru-zio-ni}
```

Nel preambolo si possono usare diverse liste introdotte mediante il comando `\hyphenation`; i loro contenuti vengono raccolti tutti assieme in una apposita memoria del programma; esso consulta sempre questa lista prima di attivare l'algoritmo di sillabazione vero e proprio.

Con lingue che usano la declinazione (come succede in parte per l'italiano) e la coniugazione, queste liste potrebbero diventare immense. In italiano un sostantivo richiede due voci: singolare e plurale; per un aggettivo ne possono essere richieste quattro: singolare e plurale, maschile e femminile; per un verbo occorrono una sessantina di voci. In italiano si ricorre assai raramente a questo meccanismo, ma, per esempio, per l'inglese ogni installazione del sistema TEX prevede la lettura automatica iniziale di una lista di eccezioni di diverse centinaia di parole.

25.3 Come fa il programma a dividere in sillabe

Tuttavia, superati questi punti, vediamo come fa il programma di composizione a dividere in sillabe.

Esso spezza ogni parola che si trovi verso la fine di ogni riga nelle sue componenti di una sola lettera, di due lettere, di tre lettere, eccetera, ma con un procedimento sequenziale che non comporti tutte le possibili scomposizioni; nel fare questo esso è guidato dai 'pattern di sillabazione'. Questi sono contenuti in un file che in fase di costruzione del formato vengono allocati in una struttura dati molto veloce da esplorare; è per questo che bisogna sempre essere sicuri che detto file sia stato elaborato nella costruzione del file di formato, altrimenti la sillabazione per la lingua cui corrispondono quei pattern è impossibile. Il programma lualatex, invece carica nel file di formato solo le impostazioni per l'inglese e carica al momento i pattern di ciascuna lingua che viene effettivamente usata nel documento; certo, se i pattern specifici di una lingua non esistono lualatex non può caricarne i pattern e tratta la quella lingua come se fosse la pseudolingua chiamata *nohyphenation*.

I pattern sono formati da monconi di parole in cui è indicata la possibilità di dividere in sillabe mediante delle cifre pari (non dividere) o dispari (dividi pure) che possono andare da 0 a 9. Se fra le stesse due lettere di pattern diversi compaiono cifre diverse, non si tratta di una contraddizione, ma fra le due indicazioni prevale quella con la cifra più alta. Per esempio, il pattern `1b` dice che si può dividere prima della 'b', ma non dopo (la cifra assente implica la cifra zero); ma il pattern `2bb` dice che si può andare a capo *dopo* la prima 'b', ma *non* prima della prima 'b'. Infatti, tenendo conto dei due pattern, il secondo si legge come se fosse `2b1b0`, il che conferma quanto appena descritto a parole.

.	d	e	l	l	’	i	s	t	r	u	z	i	o	n	e.
.	d	0 e	0 l	0 l	0 ’	0 i	0 s	0 t	0 r	0 u	0 z	0 i	0 o	0 n	0 e.
		1 l			’ 2		1 s	2	1 r		1 z			1 n	
			1 l					1 t							
		2 l	0 l					t	2 r						
.	d	0 e	2 l	1 l	0 ’	2 i	1 s	2 t	2 r	0 u	1 z	0 i	0 o	1 n	0 e.

Tabella 25.1: Pattern usati per dividere in sillabe *dell’istruzione*

Prendiamo un esempio più complesso, per esempio la ‘parola’ con elisione e apostrofo: *dell’istruzione*. Conviene rifarsi alla tabella 25.1; in questa tabella la parola iniziale è scomposta nelle sue componenti ed è preceduta e seguita da due segni di punto; questi rappresentano per il programma i delimitatori della stringa di lettere che lui ritiene essere una parola. Nella seconda riga tutti i codici di cesura pari al valore di default ‘zero’ sono intercalati alle lettere. Nella terza e nella quarta riga sono indicati i pattern che coinvolgono una sola lettera, contornati da rispettivi codici di consenso o interdizione della cesura; come si vede questi pattern coinvolgono solo le consonanti e l’apostrofo. Nella quinta riga sono riportati i pattern che coinvolgono due lettere insieme ai rispettivi codici di consenso e interdizione della cesura; per l’italiano non esistono altri pattern di tre o più lettere che si possano trovare nella parola indicata, quindi non ci sono altre righe da scrivere nella tabellina. Perciò nell’ultima riga, oltre alle lettere componenti, sono riportate le cifre più alte che compaiono in ogni colonna di cifre. Ne consegue che le cesure possono cadere solo fra le due ‘l’, tra la ‘i’ e la ‘s’, tra la ‘u’ e la ‘z’ e tra la ‘o’ e la ‘n’, dove compaiono i codici dispari.

La tabella 25.2 mostra un esempio ancora più complesso; non lo si commenta, perché il procedimento esposto in relazione alla tabella 25.1 è il medesimo. Questa volta la parola da dividere in sillabe è *discinesia*, che i medici pronunciano con il prefisso ‘dis’ separato dalla parola ‘cinesia’. Il risultato infatti porta alla divisione etimologicamente corretta *dis-ci-ne-sia*. Il lungo pattern che compare nella quinta riga, serve appunto per specificare l’eccezione alla regola di non separare la ‘s’ impura (che in questo caso, insieme alla ‘c’ e alla ‘i’ formerebbe un ‘trigramma’ che normalmente si pronuncia diversamente dalle singole lettere componenti); il pattern è sufficientemente lungo per isolare la radice di ‘discinesia’ e dei suoi derivati, come per esempio ‘discinetico’, ma non coinvolge altre parole che cominciano nello stesso modo, per esempio ‘disciplina, ‘discinto’, eccetera.

Per l’inglese i pattern sono quasi 5000, le eccezioni introdotte con il comando `\hyphenation` sono alcune centinaia e la probabilità di una divisione errata è bassa, ma non nulla.

In italiano i pattern sono circa 330, la lista delle eccezioni è nulla e negli ultimi 20 anni non sono stati segnalati errori; tutto ciò la dice lunga sulla facilità della divisione fonetica dell’italiano e della difficoltà della divisione ritmica dell’inglese; in questa lingua due parole ortograficamente identiche ma che si pronunciano con accenti ritmici diversi, richiedono divisioni diverse: *record* (nome) e *record*

.d	i	s	c	i	n	e	s	i	a.
.d 0	i 0	s 0	c 0	i 0	n 0	e 0	s 0	i 0	a.
	1 s 2			1 n		1 s 2			
		1 c							
.d 0	i 2	s 3	c 0	i 0	n 0	e			
.d 0	i 2	s 3	c 0	i 1	n 0	e 1	s 2	i 0	a.

Tabella 25.2: Pattern usati per dividere in sillabe *discinesia*

(verbo) si dividono rispettivamente in *rec-ord* e *re-cord*!

I pattern per l'italiano sono tradizionalmente contenuti nel file `ithyph.tex`, ma le recenti estensioni all'uso dei font codificati secondo la codifica UNICODE ha portato alla creazione di un altro sistema di file per la definizione dei pattern per le varie lingue. Secondo questo nuovo schema, il caricamento dei pattern è suddiviso fra tre file, il primo, `loadhyph-it.tex`, è semplicemente un *loader* che esegue alcune definizioni e poi carica i file dei pattern veri e propri; questi ultimi per l'italiano si chiamano `hyph-quote-it.tex` e `hyph-it.tex`; il primo riguarda i pattern relativi all'apostrofo che in codifica ASCII è lo stesso carattere usato per la virgoletta alta semplice, mentre in codifica UNICODE si tratta di due caratteri distinti; il secondo contiene i pattern veri e propri della lingua italiana. La lista vera e propria di pattern per l'italiano non cambia dal vecchio al nuovo formato, ma per altre lingue ci possono essere diversi cambiamenti legati alla presenza di caratteri specifici di quella lingua che, se fossero scritti con la codifica interna di TEX per i numeri esadecimali, oppure fossero codificati con numeri decimali, risulterebbero del tutto incomprensibili; il *loader* allora definisce quei caratteri con delle sequenze di controllo, per esempio `\ae`, al posto di `^^^^00e6`, in modo da individuare direttamente la legatura 'æ'.

Esercizio 25.1 Il lettore cerchi nell'elenco dei pattern (nel file `hyph-it.tex`) quelli che corrispondono alle possibili divisioni di una parola a sua scelta e controlli a mano la divisione prodotta dai pattern; successivamente inserisca la stessa parola come argomento del comando `\showhyphens` e verifichi se LATEX divide come egli ha trovato a mano. Per semplicità usi la classe *`article`*.

Esercizio 25.2 Perché la lista dei pattern `hyph-quote-it.tex` assegna un codice strano all'apostrofo e poi la lista dei pattern contiene pattern con uno o due apostrofi?

Esercizio 25.3 Il file dei pattern non contiene nulla per dividere in sillabe in modo etimologico la parola *dispepsia*, anche perché il dizionario Garzanti indica solo la divisione *di-spep-sia*.

Quale pattern bisognerebbe aggiungere a quelli esistenti per dividere questa parola e i suoi derivati, come per esempio *dispeptico*, in modo etimologico?

25.4 Bruttezza residua

In alcune circostanze, pur dividendo in sillabe le parole in fin di riga, rimangono delle righe sporgenti fuori della giustezza oppure troppo 'brutte' perché lo spazio interparola è stato allargato troppo.

Non è il caso di preoccuparsi troppo se l'indice di bruttezza (*badness* in inglese) è di qualche centinaio o di poche migliaia; il messaggio di bruttezza, che viene esposto sullo schermo e trascritto nel file `.log`, comincia a diventare preoccupante se si supera il valore 5000, ma certamente la bruttezza è 'infinita' se il messaggio riporta una bruttezza di 10 000.

Per la bruttezza delle righe si può specificare nel preambolo un limite sotto il quale il programma non deve preoccuparsi; il parametro `\hbadness` può venire impostato nel preambolo ad un valore diverso a seconda del tipo di documento e del suo contenuto; per esempio

```
\hbadness=5000
```

permette di eliminare tutti i messaggi relativi alle righe la cui bruttezza sia inferiore al valore 5000.

Fin qui si sta parlando della bruttezza relativa alle righe tropo "vuote" la cui bruttezza sia inferiore al valore specificato. La bruttezza è presente anche quando le righe sono troppo piene, al punto di uscire dalla giustezza. Per ridurre i messaggi relativi a queste righe si potrebbe specificare un valore per la lunghezza `\hfuzz`, per esempio:

```
\hfuzz=1pt
```

cosicché non ci siano messaggi per righe che "forano" la giustezza di meno di 1 pt. Quando si usa il pacchetto *microtype* è difficile che questo succeda ma non è impossibile; quindi non è fuori luogo specificare un buon valore anche se è ammesso un certo grado di protrusione dei caratteri quando si usa questo pacchetto. Ricordiamo che un punto tipografico equivale a circa un terzo di millimetro; l'occhio lo vede benissimo, ma generalmente non ne è disturbato.

In modo analogo si possono eliminare alcuni messaggi per la bruttezza verticale, che TEX emette quando deve allargare troppo la gomma dei contrografismi verticali, durante la composizione della pagina; il parametro specifico si chiama `\vbadness`; talvolta la bruttezza verticale può essere notevole, ma, a parte stabilire un valore per il parametro appena menzionato, se si vuole rendere la composizione veramente professionale bisogna apportare modifiche al testo da comporre. Generalmente questa bruttezza verticale dipende dal fatto che un oggetto ingombrante (come ad esempio il titolino di una sezione seguito da almeno due righe di testo) non trova posto al fondo di una pagina. Allungando il testo che precede l'oggetto ingombrante si riduce l'ammontare dell'allargamento della gomma verticale. Analogamente al caso delle righe che sforano la giustezza, si può specificare un valore `\vfuzz` per lo sforamento dell'altezza della gabbia sotto alla quale il programma di composizione non deve emettere messaggi.

Tornando alle righe brutte si può procedere come per le pagine brutte: si modifica il testo. Tuttavia sarebbe meglio conoscere il meccanismo con cui il programma divide i capoversi in righe. Siccome però questo argomento si addentra nelle parti più tecniche, si rinvia il lettore al testo di base, il TEXbook, dove l'argomento è sviscerato nei dettagli. Una breve descrizione di questo meccanismo è riportata nel capitolo 27.

Qui il lettore si accontenti di sapere che la brutta divisione che il programma ha trovato è quella che rende minima la bruttezza dell'intero capoverso; dunque c'è poco da fare per ridurre la bruttezza, visto che questa è la minima possibile. Si può tentare, per esempio, di introdurre degli altri punti di cesura mediante il comando `\-`; ma si veda nel seguito che cosa potrebbe succedere.

25.5 I pattern per la lingua italiana

I pattern della lingua italiana si possono creare semplicemente implementando le regole grammaticali specificate in qualunque grammatica, salvo che le grammatiche generalmente si riferiscono alla lingua conosciute dai bambini delle elementari e delle medie; generalmente non prevedono l'uso di termini tecnici e/o di parole italiane basate su una radice straniera. Le regole, comunque sono queste.

1. Ogni sillaba contiene una vocale o un dittongo o un trittongo; un dittongo è formato da due vocali di cui almeno una sia una 'i' o una 'u' non tonica[2]; un trittongo è formato da tre vocali di cui una sia 'a' oppure 'e' oppure 'o' e le altre due siano delle 'i' o 'u' atone, per esempio 'uie', 'iuo', 'ieu', 'iau', 'uoi', eccetera. Quattro vocali di seguito contengono sempre un trittongo oppure due dittonghi. Le vocali di un dittongo o di un trittongo fanno parte di un'unica sillaba.

2. Ogni consonante non viene mai separata dalla vocale che la segue. Di conseguenza in ogni parola che cominci con un gruppo di consonanti, queste fanno parte della sillaba che contiene la prima vocale che le segue; in una parola che termini con una o più consonanti, queste formano sillaba con la vocale che le precede.

3. Un gruppo di due o più consonanti in posizione intervocalica possono essere separate fra due sillabe adiacenti nel punto più a sinistra che lasci alla

[2]Queste vocali che precedono o seguono altre vocali, senza essere toniche funzionano come se fossero delle consonanti, e non si staccano dalle vocali che esse precedono (dittonghi ascendenti) o da quelle che seguono (dittonghi discendenti) e vengono spesso classificate come semivocali. I gruppi 'ui' e 'iu' sono un po' particolari e talvolta formano degli iati, in particolare il primo gruppo; se esso è preceduto da una 'q', la 'u' è sempre semivocale; se è preceduto da una 'c' il gruppo è uno iato. Tuttavia esistono parole in cui gli stessi suoni sono scritti con la 'c' o con la 'q' e non si sa perché se non se ne conosce l'etimologia, per esempio 'obliquo' e 'proficuo'; la pronuncia di quelle terminazioni è uguale e non è evidente perché una contenga la 'q' e l'altra la 'c', se non se ne conosce l'etimologia. In altri casi, come circuito, la parola viene pronunciata in modo diverso a seconda che ci riferisca al participio passato di circuire, o al nome comune. Per questo tutti gli iati ai fini dei pattern vengono trattati come se fossero dei dittonghi.

sua destra un gruppo di consonanti che possa trovarsi all'inizio di una parola italiana; per questo motivo le consonanti doppie si dividono fra due sillabe adiacenti, così come il gruppo 'cq', perché non si trovano mai all'inizio di una parola italiana. Per questo stesso motivo la consonante 's' non si divide mai dalle consonanti (diverse da 's') che la seguono; per questo stesso motivo le consonanti liquide ('l' ed 'r') e nasali ('m' ed 'n') si staccano sempre dalle consonanti che le seguono; sempre per questo motivo non si dividono i gruppi 'gl' e 'gn'.

4. È consentita, ma non è obbligatoria, la divisione etimologica anche se viola le regole precedenti.

La regola 3 è la più discutibile, anche se può essere applicata tranquillamente scrivendo in italiano di registro semplice, come quello usato dai bambini, ma non prende in considerazione diverse parole che possono essere presenti in un registro elevato.

La parola 'wagneriano' va divisa in *wag-ne-ria-no* anche se viola la regola suddetta; lo stesso vale per *watt-me-tro, mass-me-dio-lo-go* e altre simili parole composte; lo stesso vale per *tay-lo-ri-smo, new-to-nia-no, new-yor-ke-se, max-wel-lia-no, leish-ma-nio-si* e con simili parole con radice straniera.

Ma anche la regola delle consonanti iniziali di parole italiane non è tanto valida; come ci si comporta con i gruppi di consonanti iniziali di parole italiane come *bdelio, cnidio, ctenidio, ftaleina, gmelinite, pneuma, psiche, pteridina, tmesi*? La parola 'istmo' si divide *ist-mo* o *is-tmo* visto che esiste la parola italiana 'tmesi' che comincia con 'tm'? D'accordo, sono parole italiane rare e molto tecniche, tutte di origine greca, ma come si fa ad enunciare la regola 3 per escludere le parole di origine greca? Che cosa ne sa la maggior parte delle persone dell'etimologia delle parole? Certo si può consultare un vocabolario, ma molte di quelle parole mancano da quasi tutti i vocabolari. Inoltre sono pochi i vocabolari che indicano la sillabazione delle parole; fra i pochi che lo fanno il Garzanti indica la divisione *a-pne-a* rispettando la regola come enunciata in 3, ma se si escludono le parole di origine greca, la divisione diventa *ap-ne-a* molto più naturale e conforme alla pronuncia dell'italiano. Ma. . . : 'apnea' è una parola composta, quindi è valida anche la divisione etimologica *a-pne-a*.

Ci sono poi altre osservazioni da fare in merito ai gruppi di vocali. Infatti i pattern per l'italiano non dividono i dittonghi (ed è giusto), ma non dividono nemmeno gli iati; questa decisione è stata presa con una motivazione 'psicologica': il lettore si trova a disagio se trova un gruppo di due o più vocali spezzato fra due righe, anche quando la grammatica lo consentirebbe; si è deciso di spezzare i gruppi di tre o più vocali se queste contengono almeno un dittongo o un 'trittongo', ma lasciando un eventuale trittongo indiviso; quindi 'quieto' si divide solo in 'quie-to' perché contiene un trittongo, ma 'maieutica' si divide in 'ma-ieu-ti-ca' perché le quattro vocali contengono un trittongo. Però 'aiuola' viene diviso solamente in 'aiuo-la' e non in 'a-iuo-la', come la grammatica consentirebbe, perché il programma non divide dopo sillabe iniziali o prima di sillabe finali 'troppo corte'.

Una sillaba iniziale o finale è troppo corta se contiene meno lettere del numero specificato dai parametri `\lefthyphenmin` e `\righthyphenmin`; per l'italiano entrambi questi parametri valgono 2, mentre per l'inglese, il francese ed altre lingue essi valgono rispettivamente 2 e 3.

Con i vincoli tipografici e psicologici indicati, pertanto, parole come 'idea', 'maestro', 'eroe', 'croato' perdono dei possibili punti di sillabazione, ma rispettano le regole della cesura. All'occorrenza è sempre possibile intervenire a mano mediante l'uso del carattere attivo `"`, oppure mediante la predisposizione di elenchi di parole divise da dare in pasto al comando `\hyphenation`; chi scrive, però non ne ha mai avuto bisogno, specialmente da quando i programmi di composizione del sistema TEX possono usare il pacchetto *microtype* che consente una composizione migliore e con pochissime parole divise in sillabe in fin di riga. Questo testo ha pagine intere in cui non compare nemmeno una cesura o dove compare una cesura sola.

25.6 Cosa fare con le righe troppo lunghe

Perciò in quelle rare circostanze in cui la giustezza non consente la composizione perfetta nemmeno usando *microtype*, per un dato capoverso diviso in modo brutto, si potrebbe formare un gruppo dove si impostano i due parametri `\lefthypHenmin` e/o `\righthyphenmin` a valori più bassi e/o si inseriscono le divisioni esplicite, per esempio, di `pa"e"se` oppure di `i"de"a"le`. La bruttezza si sposta dalla forma del capoverso alle cesure scelte in modo grammaticalmente corretto ma tipograficamente infelice.

Un'altra possibilità è quella di usare l'ambiente *sloppypar* con le impostazioni e la sintassi seguente:

```
\begin{sloppypar}\tolerance=⟨tolleranza⟩
⟨capoverso da comporre⟩
\end{sloppypar}
```

Il valore ⟨*tolleranza*⟩ viene specificato mediante un numero inferiore a 10 000, ma altrimenti piuttosto alto; chi scrive usa talvolta il valore 9999. Si tratta di una forte tolleranza, che però, secondo il programma, non è infinita. Il capoverso avrà una certa bruttezza, ma non sarà infinitamente brutto.

La soluzione migliore, se il contenuto del capoverso lo consente, è però quella di modificare il testo; spesso basta aggiungere o togliere una parola, oppure scambiare di posto due parole, oppure riformulare un periodo o una frase. Se il capoverso non è troppo corto le possibilità sono innumerevoli.

Il lettore attento avrà notato che in questo testo le cesure in fin di riga sono rarissime; in parte è merito dell'uso del pacchetto *microtype* e in parte del fatto che la giustificazione delle righe mediante il sapiente allargamento e restringimento dello spazio interparola, guidato dall'algoritmo di minimizzazione della bruttezza di ciascun capoverso, rende la divisione in sillabe piuttosto rara.

Questo è uno dei pregi dei programmi di composizione tipografica del sistema TEX che li rendono insuperabili in molte applicazioni.

25.7 I file di pattern

Con le ultime distribuzioni di TEX Live e di MiKTEX non sarebbe il caso di preoccuparsi delle lingue con le quali si può dividere in sillabe un testo scritto in una delle moltissime lingue che il programma sa gestire tramite il pacchetto `babel` o il pacchetto `polyglossia` e i loro vari file di descrizione. L'ultima volta che ho controllato (2016), le lingue gestibili erano poco più di 80; ora (2021) sono più di 100.

Con le distribuzioni di base o ridotte al minimo indispensabile, bisogna leggere la documentazione che accompagna la propria distribuzione per sapere come fare per attivare quelle lingue che non sono attivate in prima installazione.

Generalmente con TEX Live non c'è bisogno di preoccuparsi perché l'installazione completa è quella di default, e per eseguire un'installazione parziale bisogna mettercela tutta per specificare le opportune opzioni al programma di installazione; non è una operazione semplice, quindi gli utenti che installano TEX Live la installano quasi sempre completa, e fanno benissimo.

Invece possono incontrare problemi coloro che usano la distribuzione TEX Live per una macchina Linux conforme alle prescrizioni del consorzio Debian. A causa di queste prescrizioni e della macchinosa procedura di accettazione del software conforme alle specifiche Debian, questa distribuzione, anche se installata di default con molte varianti del sistema operativo Linux, è sempre un po' datata, oggi meno rispetto a qualche anno fa quando poteva essere "in ritardo" di aggiornamento anche di un paio d'anni. Nello stesso tempo quella installazione obsoleta e fortemente parziale è necessaria per soddisfare le dipendenze di molti altri programmi destinati ad un sistema Linux conforme a Debian. C'è però il modo di aggirare questo problema in modo indolore e perfettamente corretto per la sicurezza della machina. Infatti due distribuzioni TEX Live possono tranquillamente convivere nella stessa macchina senza disturbarsi a vicenda; si veda il manualetto [27] dove è specificato come fare per installare una versione di TEX Live completa accanto alla versione parziale Debian.

Chi usasse una distribuzione MiKTEX di base scoprirebbe che questa in prima installazione è predisposta per gestire la sillabazione di una mezza dozzina di lingue, fra le quali non compare l'italiano. L'utente deve allora attivare dal menù Start/Avvio la pagina di dialogo dei "MiKTEX Settings"; deve cliccare sulla linguetta "Languages" dove marca con l'apposito segno le lingue di cui desidera poter usare la sillabazione (un'ottima idea è quello di marcarle tutte) e, dopo aver cliccato sul pulsante "OK", deve selezionare la linguetta "General" ove cliccherà sul pulsante "Regenerate all formats". Questa semplice operazione (ripetibile qualora non si siano marcate tutte le lingue disponibili e se ne vogliano aggiungere altre) consente di dimenticarsi del problema dei pattern di sillabazione perché a quel punto MiKTEX è in grado di gestire tutte le lingue che conosce.

Ma se l'utente stesse scrivendo un testo sull'ostrogoto altomedievale? Ammesso che disponga dei font adatti e si sia scritto un file di descrizione di questa lingua (sulla falsa riga dei file `italian.ldf`, `latin.ldf`, eccetera), in modo da disporre di tutto il necessario per comporre in ostrogoto, scoprirebbe che il suo testo è diviso in sillabe con le regole di default, vale a dire con le regole dell'*inglese americano* oppure non è diviso affatto se compone con il programma lualatex.

Già! Mancano i pattern. Ma dove sono o dove trova l'utente i pattern dell'ostrogoto altomedievale? Semplicemente non esistono (almeno a tutt'oggi, 2019).

Se ha abbastanza pazienza e conosce bene la lingua ostrogota altomedievale (si suppone che lo sia, visto che sta scrivendo un testo in quella o su quella lingua) può crearsi i pattern da solo. Chi scrive l'ha fatto fin dall'inizio della sua storia con LaTeX, e a suo tempo produsse la prima versione dei pattern per l'italiano, oltre che per diverse altre lingue romanze, per il greco (classico e moderno), e per il copto (ma molto più recentemente)[3]; questo non vuol dire che egli conosca a perfezione tutte le lingue romanze e il greco classico nonché quello moderno, ma si era documentato molto bene in merito ed era riuscito nell'intento. In seguito molti utenti del sistema TeX di madrelingua intervennero per predisporre pattern migliori, tenendo conto che nel frattempo il motore di composizione era diventato la prima approssimazione di π e che si erano resi disponibili i font latini con codifica *T1*. Per il greco gli utenti greci di TeX si erano dati un gran da fare e in pochi anni, dopo la disponibilità generale dei font greci con codifica *LGR*, hanno realizzato degli ottimi file di pattern; `babel` carica tre diversi file di pattern uno per il greco monotonico e gli altri due per le due forme di greco politonico. Il programma xelatex, con `polyglossia`, e i rinnovati file di pattern

[3] Cominciò verso la fine degli anni '80, quando il programma di composizione tex gestiva ancora una sola lingua alla volta, e non esistevano i font con codifica *T1*. I francesi avevano risolto il problema creandosi una variante della collezione dei font Computer Modern a 128 caratteri, ma che contenesse anche le lettere accentate e gli altri segni speciali che compaiono in francese, per cui avevano anche definito un sistema di pattern che facesse riferimento a quel tipo di font. In Italia non disponevamo di questa variante del font generata dagli utenti francesi, quindi gestire le lingue europee con accenti, vincolati ai font Computer Modern, era un problema non indifferente, che però si riusciva a risolvere giocando fra pattern e macro.

Per l'italiano non esistevano grossi problemi perché comunque l'accento obbligatorio cade solo sull'ultima vocale delle parole tronche; per spagnolo, catalano e portoghese ci si poteva comportare come per il francese giocando fra macro e pattern; per il latino normale (non rinascimentale e non ecclesiastico) non c'erano problemi salvo il fatto che per il latino classico ci sarebbe voluta la sillabazione etimologica, non fonetica, e si riuscì a risolvere il problema con un compromesso fra l'etimologico e il fonetico. Per il rumeno non fu un grande successo, visto che la presenza di segni speciali è molto maggiore che nelle lingue romanze occidentali, tuttavia anche in quel caso si riuscì ad ottenere un risultato decoroso. Col greco non ci furono problemi particolari, senonché nel desiderio di avere un solo file di pattern per il greco antico politonico, per il greco moderno politonico e per il greco moderno monotonico, chi scrive aveva optato per una soluzione molto conservativa: *in dubio abstine*, e i pattern erano molto precisi, nel senso che sbagliavano la cesura molto raramente, ma mancavano molti punti leciti di cesura. Da allora gli unici pattern rimasti dalla prima stesura, sia pure abbondantemente riveduti e corretti, sono quelli per l'italiano e il latino; le due versioni per il copto arrivarono quasi dieci anni dopo.

per renderli compatibili con la codifica *utf8*, carica invece tutti e tre i file di pattern a seconda dell'opzione specificata per la lingua greca. Usando lualatex, ad oggi, ottobre del 2019, non si è ancora superato il vincolo di caricare un solo file di pattern per ciascuna lingua, quindi per il momento questo programma si serve unicamente dei pattern per il greco monotonico (cosa che per altro soddisfa la maggior parte degli utenti greci, ma non soddisfa affatto gli ellenisti di ogni nazionalità che scrivono con, o di, greco politonico antico e moderno). Questo problema esiste anche per le varie forme di latino, ma il problema è in via di risoluzione.

Nel 2014 il modulo di `babel` per il latino è stato aggiornato e può fare uso di pattern distinti per il latino moderno e per quello medievale e rinascimentale basati su divisioni in sillabe su base prevalentemente fonetica, oppure per il latino classico che ha la sillabazione basata prevalentemente su regole etimologiche. Per `polyglossia` le tre varianti sono disponibili dalla versione 2015 di TEX Live quando si usa xelatex, mentre sono in via di definizione per lualatex.

Questo *excursus* storico rende bene l'idea del lavoro che c'è stato dietro la produzione del necessario per comporre in un centinaio lingue diverse. Ciò nonostante può essere necessario produrre ulteriori file di pattern. Per esempio, nella prima decade di questo secolo fu chiesto a chi scrive di predisporre i font e i pattern per comporre nella varietà di copto, detta *bohairico*, usata nella liturgia della chiesa copta. Su richiesta di alcuni utenti, nella prima metà della seconda decade di questo secolo sono stati predisposti i pattern per comporre in retoromancio, in friulano, in piemontese e in occitano. Con la distribuzione di TEX Live del 2021 dovrebbero diventare disponibili anche i pattern per l'albanese.

I pattern sono molto legati alla codifica dei font di uscita; i font per il copto bohairico non trovano posto nella codifica UNICODE, ma sono traslitterabili tranquillamente con i segni della tastiera latina; i pattern furono fatti, ma per ora vengono usati solo da chi scrive e da un piccolo gruppo di persone, diaconi della chiesa copta di Gran Bretagna, che stanno ricomponendo i libri liturgici, specialmente per i fedeli inglesi o naturalizzati, con il testo copto che procede parallelamente al testo inglese. Chi scrive non ha avuto difficoltà particolari nella creazione dei pattern, avendo ricevuto le relative istruzioni grammaticali da uno dei diaconi coinvolti nel progetto e con la sua verifica periodica della correttezza dei risultati.

Questi pattern sono stati caricati anche su CTAN e fanno parte di TEX Live, anche se sono funzionali solo per comporre testi di liturgia copti.

Per comporre in copto bohairico bisogna specificarne il nome con il comando di attivazione del pacchetto babel, come si fa per le altre lingue e bisigna usare i font adatti, già presenti nell'installazione aggiornata e completa di TEX Live.

Quanto fatto per il copto si può ripetere anche per l'ostrogoto altomedievale[4];

[4]Ho inventato il nome 'ostrogoto altomedievale' tanto per fare un esempio di una lingua forse esistita, ma improbabile; tuttavia recentemente uno dei collaboratori di questa guida mi ha chiesto se me la sentivo di predisporre il necessario per comporre in *cimbro*; non ci ho ancora messo mano, anche perché non è possibile trovare una grammatica che specifichi le regole di sillabazione di questa lingua, almeno non l'abbiamo ancora trovata. Non solo,

l'importante è che il nostro ipotetico scrittore sappia costruirsi il file di pattern o sappia a chi rivolgersi per creare detto file. Si ricordi che ad ogni modifica, aggiunta o correzione del file di pattern, deve ricreare il formato almeno del programma che usa per comporre il suo testo, presumibilmente pdflatex.

Esiste un modo per creare il file di pattern senza dover conoscere la grammatica della lingua? Sì, esiste, ed è quello che ha usato Knuth o meglio, il suo collaboratore Frank Liang, per generare i pattern in modo automatico; 'automatico", per la verità, è una parola grossa; bisogna disporre o predisporre un elenco di parole, tanto più numerose quanto più sono le eccezioni ad una regola precisa, già divise in sillabe, e da dare in pasto al programma patgen, che fa già parte della distribuzione TEX Live. Purtroppo la documentazione, leggibile con `texdoc patgen` dal terminale, riguarda il programma patgen in quanto tale; come si usi è descritto in modo molto parziale e comprensibile solo a chi ci mette molta buona volontà. Tuttavia la vera difficoltà non risiede nel comprendere come si usi patgen, ma consiste nel generare il file sorgente di parole già divise in sillabe e traslitterate con l'alfabeto latino anche con le lettere accentate secondo la codifica *T1*. Si capisce benissimo che i pattern generati saranno tanto più efficaci quanti più casi di sillabazione sono presenti nel file di ingresso per patgen. Per i pattern validi con la codifica *utf8* la procedura è simile; richiede di usare una variante di patgen adatta per gestire i caratteri Unicode, ma solitamente ci pensano i membri del gruppo di lavoro del TUG, che si occupano della cesura, a trasformare in pattern per font "one-byte" (da usare con pdflatex) in pattern per font "multi-byte" (da usare con xelatex e lualatex) o viceversa.

Frank Liang ha trovato in rete inizialmente un elenco di 25 000 parole in inglese americano già sillabate e su quello ha costruito patgen; verificatane l'efficienza e la mancanza di errori di programmazione, è poi riuscito ad ottenere un elenco di 100 000 parole con cui ha costruito i primi pattern per l'inglese americano; ed è lui che ha segnalato che l'elenco dei quasi 5000 pattern ottenuti esegue correttamente la sillabazione nel 90% dei casi.

Negli anni successivi sono stati elaborati altri file di pattern, distinti per l'inglese americano e per quello britannico, i quali contengono un numero molto maggiore di pattern, ma contemporaneamente il TUG internazionale tiene aggiornato un elenco di eccezioni che fa parte di ogni aggiornamento di TEX Live.

Ora l'efficienza della sillabazione per l'inglese è molto aumentata, ma non può evidentemente distinguere le parole omografe che si pronunciano diversamente. La sillabazione fonetica dell'italiano è molto più semplice perché, anche se pronunciate diversamente, le parole omografe sono sillabate nello stesso modo. Il problema degli omografi si presenta con molte altre lingue; in latino, per esempio, 'languere' è sia l'infinito del verbo sia una alternativa al perfetto 'languerunt'; la scrittura prosodica del latino li distinguerebbe: 'languĕre' e 'languēre', ma il latino normalmente si scrive senza diacritici, comunque senza quelli prosodici. Il

ma la mia conoscenza delle lingue portate in occidente dai goti è molto scarsa, per non dire totalmente assente; conosco pochissimo il tedesco e riconosco gli scritti in lingue alemanniche, ma una cosa è riconoscere uno scritto, un'altra completamente diversa è conoscere la lingua.

latino ecclesiastico e quello liturgico potrebbero distinguere i due omografi con 'lànguere' e 'languére'.

Chi scrive ha sempre lavorato a mano per generare i file di pattern e ha usato patgen per vedere se con l'italiano si potevano ottenere meno pattern dei 330 circa che egli ha usato. In realtà il numero di pattern con patgen era marginalmente maggiore di quelli ottenuti a mano ma il numero di "trie" era decisamente maggiore, anche perché i pattern generati con patgen arrivavano fino al codice di consenso 5, mentre quelli generati a mano arrivavano solo fino a 4. Il punto vero è che non ho la più vaga idea di quanto debba essere grande un "dizionario" per le parole italiane già divise in sillabe e non ho idea di quanto tempo occorra per scriverlo e controllare che sia assolutamente privo di errori. Ma disponendo di un tale dizionario, lavorare con patgen può, forse, non dare luogo al miglior risultato possibile ma, disponendo del "dizionario" di parole già divise in sillabe, il lavoro per creare i pattern con patgen si completa in meno di un paio d'ore.

Capitolo 26

Codifiche in entrata e in uscita

Il problema delle codifiche è delicato e spesso difficile da capire da parte di chi non conosce il funzionamento interno del proprio calcolatore. Chi vuol arrivare subito alle conclusioni operative salti direttamente all'ultimo paragrafo di questo capitolo: 26.5. Poi se vuole capire perché vengono suggerite quelle soluzioni può tornare qui all'inizio per vedere le cose tecniche che portano a quelle conclusioni.

26.1 Introduzione

L'argomento della codifica è uno di quelli che più travagliano gli utenti del sistema TEX. Per cercare di fare un po' di luce su questi argomenti, bisogna rifarsi al calcolatore che elabora i dati. Al suo interno ogni tipo di informazione è "scritto" in termini di cifre binarie '0' e '1'. Sia ben chiaro: queste cifre non sono davvero scritte nel modo che noi umani comunemente associamo al verbo 'scrivere'. I segni '0' e '1' sono essi stessi dei modi codificati per descrivere gli stati interni di certe quantità fisiche all'interno del calcolatore: saranno tensioni oppure correnti positive o negative; saranno delle piccolissime parti di materiale magnetico magnetizzato con il polo nord in alto oppure in basso; saranno delle cariche elettriche positive o negative localizzate in certi cristalli di silicio o di altro materiale semiconduttore; in realtà non ci interessa andare così a fondo nella struttura della macchina per capire che le cifre binarie '0' e '1' sono modi convenzionali per descrivere uno o l'altro dei due stati opposti che possono assumere quelle grandezze fisiche in quei determinati punti della macchina. A noi interessa il significato che di volta in volta viene attribuito a quei modi di essere di quelle grandezze fisiche.

I segnali che vengono trasmessi da un punto all'altro della macchina sono sequenze di 'zeri' e 'uni' e l'informazione che essi trasportano da un punto all'altro della macchina dipende dal trasmettitore e dal ricevitore che usano un *codice* per

attribuire questo significato e per comportarsi di conseguenza. Ogni 'zero' e ogni 'uno' di una sequenza si chiama *bit*, un breve acronimo diventato nome comune, ottenuto da "Binary unIT", unità binaria, e rappresenta la minima quantità di informazione che può essere contenuta in un messaggio.

A livello macroscopico, quando noi battiamo un tasto sulla tastiera questa invia una sequenza di bit particolare (ovviamente diversa da tasto a tasto) all'unità centrale di elaborazione che ne farà l'uso corrispondente al ruolo che in quel momento la tastiera sta svolgendo – la tastiera non svolge solo il ruolo di strumento di scrittura. La tastiera fisica non manda all'unità centrale una lettera, ma una sequenza di bit; otto bit (oggi[1]) formano un *byte*. Un byte di otto bit può assumere 256 configurazioni diverse: 00000000, 00000001, 00000010, 00000011, ..., 11111111, associabili, come numerazione binaria ai numeri decimali 0, 1, 2, 3, ..., 255. Ma, ecco, già questa è una codifica (anche se non è l'unica lettura possibile), perché a ogni sequenza di bit viene associata una lettura in termini di numero binario; insomma con codifiche diverse (modi di interpretazione diversi) lo stesso byte può assumere significati diversi.

Ma è solo una faccia della questione che riguarda la codifica. Torniamo alla nostra tastiera: la sequenza di bit che una data tastiera invia all'unità centrale è la stessa quando si preme lo stesso tasto ma, per esempio, il secondo tasto da sinistra della terza fila dall'alto significa 'w' con una tastiera QWERTY ma significa 'z' con una tastiera QZERTY. Per facilitare la lettura agli umani, sui tasti sono disegnati determinati segni, ma la tastiera fisica è del tutto indifferente a ciò che è disegnato sui tasti. Tra la tastiera e l'unità centrale di elaborazione c'è quindi un 'filtro' che traduce la sequenza di bit inviati dal tasto premuto in un segno piuttosto che un altro a seconda di come l'operatore ha configurato la macchina quando ha risposto ad una apposita domanda che gli ha fatto il programma di installazione del sistema operativo: "che tipo di tastiera hai?"; questo è particolarmente evidente quando si installa una distribuzione Linux, perché questo sistema operativo non è legato a nessun particolare tipo di hardware, nel senso che dovrebbe essere in grado di adattarsi a qualsiasi tipo di hardware di qualunque marca e prodotto da qualunque fabbricante.

Il filtro, comunemente chiamato *driver* di tastiera, può però essere cambiato a piacere; l'utente può installare diversi driver di tastiera sulla sua macchina, e li può rendere attivi (uno alla volta, evidentemente) in modo da poter toccare gli stessi tasti, ma inviare il segno 'w' oppure il segno σ, o qualunque altro segno di qualunque alfabeto che il driver faccia corrispondere a quel famoso secondo tasto da sinistra della terza fila dall'alto. Si tratta di codifiche diverse: cambiando driver si cambia il modo di interpretare la stessa sequenza di bit.

Nello stesso modo possiamo distinguere una codifica d'ingresso quando l'unità centrale di elaborazione (CPU, *central processing unit*) e il programma che in quel

[1] In passato il byte era definito come la più piccola sequenza di bit associabile a un indirizzo di memoria del calcolatore; questi "bocconi" di informazione potevano essere formati da 7, o da 8, o anche da 9 bit; potevano anche avere tre bit in più se erano dei byte autocorrettivi; ma questo esula completamente dall'argomento di questo testo. Basti sapere che "oggi" tutti i sistemi di calcolo usano byte di 8 bit.

momento sta "girando" ricevono una sequenza di bit da qualunque periferica capace di inviare alla CPU segnali formati da sequenze di bit, e una codifica di uscita quando sono l'unità centrale di elaborazione e il programma che vi sta girando in quel momento che inviano segnali alle periferiche. I concetti di 'ingesso' e di 'uscita' si riferiscono quindi ai segnali che entrano nella CPU per essere elaborati, oppure che vi escono dopo l'elaborazione per essere utilizzati da "terze parti".

Nel caso particolare del sistema TEX, ci ritroviamo con l'unità centrale sulla quale gira il programma che chiamiamo editor; i segnali di entrata arrivano dalla tastiera attraverso il suo driver o dal disco fisso (o da qualunque altro dispositivo di memorizzazione di massa) attraverso un altro driver specifico. L'unità centrale e l'editor che vi sta girando inviano segnali al video e al disco fisso (o altro dispositivo) ciascuno attraverso il suo specifico driver. L'editor opportunamente azionato può inviare segnali anche al sistema operativo per ordinargli di mettersi ad eseguire uno dei programmi di composizione del sistema TEX; la CPU sospende l'esecuzione dell'editor e lo mette da parte, carica e lancia l'esecuzione di quel programma del sistema TEX, e assieme richiedono al disco fisso di inviare all'unità centrale i segnali corrispondenti al testo da comporre con il mark-up di LATEX; il flusso di segnali viene elaborato dalla CPU e il risultato viene inviato nuovamente al disco fisso e viene chiuso il programma di composizione richiamando dal suo riposo l'editor; la CPU invia all'editor segnali che il processo di composizione è andato a buon fine, e l'editor rilancia segnali per informare l'unità centrale che può attivare il programma di visualizzazione; il gioco ricomincia: l'unità centrale sospende l'editor e lo rimette da parte, apre o riprende il programma di visualizzazione precedentemente messo da parte e gli dice di riprendere il suo lavoro; questo programma a sua volta richiede al disco di lanciargli i segnali dei dati da visualizzare e provvede con suoi driver interni a mandare i debiti segnali al video affinché esso mostri sullo schermo il risultato della composizione; eventualmente l'operatore dà ordine all'editor di mettere in stampa il prodotto della composizione, ma sebbene le macchine periferiche siano diverse, i concetti generali di collaborazione fra le varie parti della macchina, i programmi e le periferiche è sempre lo stesso.

Non continuo perciò a descrivere questo dialogo fra sistema operativo, CPU, periferiche come la tastiera, il video e i dischi e gli altri sistemi di memorizzazione di massa, la memoria volatile interna, chiamata RAM, per tenere da parte i programmi che in quel momento sono inoperosi, perché il discorso diventerebbe lunghissimo e noiosissimo. Metto in risalto che ogni volta che si ha un dialogo fra una parte e l'altra della macchina o fra un programma e l'altro, i segnali sono in codice e ogni macchina periferica o centrale può decifrarli attraverso gli appositi driver.

Che cosa c'entra allora l'operatore/compositore in tutta questa storia di codifiche? Perché il sistema operativo non provvede da solo a gestire queste cose senza tirare in ballo quel poveraccio del compositore? Con i programmi e i sistemi chiusi questo in genere succede, per esempio con la componente Writer di Libre Office questi problemi non si presentano; il sistema TEX non è un sistema

32	␣	48	0	64	@	80	P	96	‘	112	p
33	!	49	1	65	A	81	Q	97	a	113	q
34	"	50	2	66	B	82	R	98	b	114	r
35	#	51	3	67	C	83	S	99	c	115	s
36	$	52	4	68	D	84	T	100	d	116	t
37	%	53	5	69	E	85	U	101	e	117	u
38	&	54	6	70	F	86	V	102	f	118	v
39	’	55	7	71	G	87	W	103	g	119	w
40	(	56	8	72	H	88	X	104	h	120	x
41	)	57	9	73	I	89	Y	105	i	121	y
42	*	58	:	74	J	90	Z	106	j	122	z
43	+	59	;	75	K	91	[	107	k	123	{
44	,	60	<	76	L	92	\	108	l	124	|
45	-	61	=	77	M	93	]	109	m	125	}
46	.	62	>	78	N	94	^	110	n	126	~
47	/	63	?	79	O	95	_	111	o	127	

Tabella 26.1: I 95 caratteri ASCII stampabili. Il carattere ASCII 32 è lo spazio, qui reso visibile (␣) mediante il comando `\textvisiblespace`. Il carattere ASCII 127 (il novantaseiesimo della tabella) non è un carattere stampabile.

chiuso in sé stesso, e non solo perché è un sistema che deve poter lavorare su qualunque tipo di hardware, con qualunque periferica e sotto il controllo di qualunque sistema operativo, ma anche perché vuole poter essere totalmente staccato da simili situazioni contingenti affinché i suoi file siano trasportabili da una macchina all’altra anche con sistemi operativi diversi.

Il sistema TEX al suo interno funziona con ulteriori codifiche sue proprie.

26.2 Le tre distinte codifiche di TEX

Nella pagina 401 si è definito ciò che si intende per codifica di un font. La definizione si riferiva ai file relativi ai font che vengono usati per ‘scrivere’ il documento nel file di uscita. Nello stesso tempo si è parlato spesso di codifica dei caratteri con cui si scrive il file `.tex` da comporre. Quindi si è parlato di codifiche di uscita e di codifiche di entrata.

Qui bisogna fare attenzione a non confondersi; infatti la codifica in entrata, quella che serve per creare o modificare i file `.tex`, riguarda il modo di funzionare dello *shell editor* con il quale si eseguono queste operazioni e con il quale si salva il file `.tex`. La codifica in uscita riguarda le polizze dei caratteri che si usano per la composizione. Fra queste due codifiche si trova il programma di composizione che usa una sua codifica interna. Detto in altri termini: l’editor non si preoccupa del fatto che il testo e il mark-up che esso legge dal disco oppure dalla tastiera verrà successivamente elaborato da un programma invece che da un altro; l’importante

è che passi le informazioni giuste con il file che il successivo programma di composizione leggerà dal disco in modo che l'elaborazione dell'informazione contenuta nel disco produca la composizione tipografica desiderata. Ricordiamoci anche che lo *shell editor* con il quale creiamo o modifichiamo e salviamo il file `.tex` non fa propriamente parte del sistema TEX; questo interviene solo per trasformare il file sorgente `.tex` nel file composto `.pdf`.

Questa comunicazione fra l'editor e il programma di composizione si ottiene mediante l'invocazione dei pacchetti *inputenc* e *fontenc* con le opzioni appropriate.

Quando si invoca il pacchetto *inputenc* con una qualunque opzione di codifica che il pacchetto accetti, si stabilisce una corrispondenza fra la codifica dei caratteri immessi con la tastiera nello *shell editor* (che esso trascrive nel file `.tex` e che mostra graficamente sullo schermo), e quanto dovrà fare il successivo programma pdflatex per tradurlo nella sua codifica interna quando, leggendo il file `.tex`, ne eseguirà la composizione.

Quando invece si invoca il pacchetto *fontenc* con una qualunque opzione accettata da questo pacchetto, si stabilisce una corrispondenza fra il codice interno del programma di composizione e i vari segni delle polizze dei font usati, caratterizzati dalla codifica specificata e trascritti dal programma sia nella pagina in lavorazione, sia nelle pagine emesse nel file di uscita, le pagine composte che finalmente saranno visualizzate sullo schermo o stampate su carta.

Tutto ciò è particolarmente importante quando si usa pdflatex per comporre il documento in lavorazione. Usando invece lualatex o xelatex nella loro forma tipica, cioè usando i font OpenType, le codifiche di entrata, quella interna e quella di uscita sono sempre e soltanto UNICODE anche nella sua variante transcodificata UTF-8. Con questi due programmi non ci sono problemi di codifica, ma sia l'entrata sia l'uscita *devono* essere UNICODE o UFT-8, e *non bisogna specificare nessuna codifica ai file di estensione* `inputenc` *e* `fontenc`*, perché questi due file non devono assolutamente essere invocati*; solo se si chiede a questi programmi di comporre un documento predisposto per essere composto con pdflatex con qualsiasi codifica d'entrata e dei font, questi programmi si comportano come con pdflatex.

Si noti che i codici emessi dal driver della tastiera, o letti dal disco sotto forma di sequenze di bit, possono essere letti anche come numeri binari con i loro corrispondenti decimali, e noi possiamo identificare il carattere 'w', per esempio, con il numero decimale 119 e possiamo dire; "viene inviato il carattere 119 dalla tastiera all'editor". Il numero decimale corrisponde quindi ad una specie di indirizzo che corrisponde alla posizione della lettera 'w' in una certa tabella di caratteri corrispondente ad una certa codifica.

In sostanza la tastiera invia alla CPU delle sequenze di byte che le indicano quale tasto è stato premuto e se contemporaneamente erano stati premuti uno o più dei tasti di controllo, come il tasto delle maiuscole o quello delle maiuscole permanenti, il tasto etichettato Ctrl, o quello etichettato Alt, eccetera. In base alle impostazioni del sistema il driver della tastiera traduce questa sequenza

di byte in un'altra sequenza, spessissimo costituita da un solo byte, che indica l'indirizzo del segno secondo la codifica scelta. Se si tratta di un unico byte il cui primo bit sia 0, allora il carattere corrisponde ad uno di quelli indicati nella tabella 26.1, che è comune a tutte le altre codifiche, compresa la codifica UNICODE.

Se si tratta di un solo byte e il primo bit vale 1, esso può corrispondere ad un carattere diverso a seconda della codifica; nella tabella 26.2 sono riportati i vari caratteri con indirizzi compresi fra 128 e 255 corrispondenti ad alcune delle codifiche più comuni: *ansinew* adatta ai sistemi Windows (praticamente coincide con la *code page* 1252, per la quale si potrebbe specificare l'opzione *cp1252*); *latin1* adatta ai sistemi Linux e Macintosh; *latin9* adatta agli stessi sistemi ma con alcuni caratteri diversi, compreso quello dell'euro; *applemac* adatta ai sistemi Macintosh per i quali si voglia mantenere l'uso della codifica specifica, disponibile con i sistemi operativi precedenti al Mac OS X, ma usabile anche con questo sistema operativo; *cp437* corrisponde alla codifica delle macchine che usavano il sistema operativo DOS; *cp1250* corrisponde alla codifica delle macchine Windows degli anni '90 ed è la versione usata in quei paesi dell'Europa orientale che usano varianti dell'alfabeto latino.

Nella tabella alcuni caratteri sembrano non produrre nessun risultato; uno è il carattere relativo allo spazio indivisibile (*no break space*); un altro è il marcatore di parola composta (*compound word marker*) che di fatto, con il sistema TEX, è sostituito dal punto di cesura facoltativo reso con il comando `\-`.

Come si vede non c'è praticamente nessuna differenza, tranne per la colonna intestata `latin9` per soli cinque caratteri (l'euro, la Š (№ 166), la š (№ 168), Ž (№ 180), e il carattere № 255 che non viene usato) e quindi la codifica *latin1* o, se si preferisce, la codifica *latin9* possono essere usate con tutti e tre i sistemi operativi senza problemi, almeno per quel che riguarda i caratteri dal 161 al 255. Con una installazione del sistema TEX successiva al 2019 si consiglia fortemente la codifica *utf8* – vedi nel seguito; anche perché molti dei restanti segni sono tutti da rendere in uscita con la codifica speciale del pacchetto `textcomp` (che non è necessario caricare, perché le sue funzionalità sono già incluse nel nucleo di LATEX); queste funzionalità estendono il numero di glifi disponibili con un altro centinaio di simboli alfabetici e analfabetici).

Le altre colonne sono invece molto diverse dalla prime tre; lasciando perdere le ultime due colonne (che si riferiscono per lo più a situazioni particolari e che non riguardano la grande maggioranza degli utenti italiani che dispongono di macchine "moderne") la codifica *applemac* è abbastanza diffusa fra gli utenti delle macchine Macintosh. Se ne scoraggia l'uso con vigore. Scambiarsi file codificati in questo modo fra utenti che usano diverse macchine, non tutte Macintosh, vuol dire produrre fastidio e irritazione fra tutti gli autori che collaborano ad un progetto collettivo. La codifica *latin1* andava benissimo prima del 2019 per tutte le macchine aggiornate più diffuse; *la codifica utf8, di cui si parlerà fra poco, è la migliore in assoluto.*

Tanto per essere espliciti, la frase:

La città è più pulita, perché così può essere più bella.

scritta con la codifica *applemac* e aperta con un editor impostato per la codifica *latin1* appare sullo schermo così:

La citt^ pi pulita, perchŽ cos“ pu˜ essere pi bella.

Ancora migliore è la codifica UNICODE, che è quasi universale, nel senso che si riferisce a centinaia di migliaia di segni, non solo i segni delle codifiche a 256 caratteri, e comprende ogni genere di alfabeto, ogni verso di scrittura, ogni collezione di ideogrammi e di grafemi. La lista dei segni è talmente lunga che per consultarla è meglio riferirsi alle pagine UNICODE reperibili sul sito del consorzio omonimo: `http://www.unicode.org/Public/UNIDATA/UnicodeData.txt`. Nel 2022 è apparso su ArsTEXnica 33 un articolo di Frank Mittelbach che descrive un suo nuovo pacchetto, packunicodefonttable, che permette di mostrare tutti i caratteri di un dato font, anche se codificato in UNICODE e quindi può contenere migliaia di caratteri; Il pacchetto permette anche di confrontare due diversi font, evidenziando quelli che sono specifici di unno solo dei due. I codici UNICODE possono contenere fino a 6 cifre esadecimali (3 byte), di cui la prima può assumere solo i valori 0 e 1; quindi essi vanno da `000000` a `10FFFF` (cioè 1 114 112 segni; in teoria potrebbe arrivare fino a `1FFFFF` e quindi avrebbe spazio per 2 097 152 differenti segni). La codifica UTF-8 è un modo particolare, un algoritmo, per generare il codice UNICODE di ogni carattere in modo da usare meno byte per indirizzare i segni che occupano le prime posizioni della codifica UNICODE; a tutti gli effetti pratici possiamo identificare la codifica UTF-8 (*utf8* per il pacchetto *inputenc*) con la codifica UNICODE. Per saperne di più sulla codifica UTF-8 si può vedere il sito `http://it.wikipedia.org/wiki/UTF-8`. Con due byte solamente si possono indirizzare tutti i segni per gli alfabeti latino (anche con diacritici), greco, cirillico, copto, ebraico e arabo; per tutti gli ideogrammi (cinese giapponese e coreano) occorrono tre byte; UTF-8 usa quattro byte solo per segni particolarissimi che vengono usati piuttosto raramente.

In modo analogo la codifica di uscita, quella da specificare al pacchetto *fontenc*, indica la posizione che un dato carattere occupa nel file che contiene le informazioni per disegnare il suo glifo sullo schermo, o sulla carta, o sulla periferica di uscita; la codifica di uscita, quindi, riguarda i font con cui viene composto il documento finale; non per niente quando viene composto questo capoverso l'informazione che viene memorizzata nel file di uscita, prima ancora di indicare le posizioni dei caratteri del font da usare, è `\T1/lmr/m/n/10`, cioè, in codice, l'agglutinazione delle informazioni relative (*a*) alla codifica del file che contiene le istruzioni per 'disegnare' i singoli caratteri (`T1`); (*b*) alla famiglia di caratteri (`lmr`, Latin Modern Roman); (*c*) alla serie dei caratteri (`m`, serie di media nerezza); (*d*) alla forma dei caratteri (`n`, caratteri normali diritti); (*e*) al corpo (`10`, corpo di 10 pt).

Va detto che in teoria il pacchetto per la codifica d'ingresso *utf8* potrebbe codificare più di un milione di segni; però la cosa sarebbe poco pratica e probabilmente impossibile da realizzare anche con i calcolatori più moderni che possono

usare memorie di grandissime dimensioni. Per cui il pacchetto *inputenc* con l'opzione *utf8* si limita a definire solo le codifiche dei caratteri che possono venire effettivamente rappresentati con i font di uscita; per poterlo fare deve conoscere le codifiche di tutti i font che verranno usati nel corpo del documento. Per questo sarebbe opportuno caricare il pacchetto *inputenc* solo dopo aver caricato tutti gli altri pacchetti che si riferiscono ai font. In particolare sarebbe opportuno caricare prima, non solo il pacchetto *fontenc* con le opzioni che si desiderano, ma anche ogni altro pacchetto per l'uso di collezioni di simboli speciali non compresi in quelli rappresentabili con la codifica *T1* né con la codifica *TS1* ora definita nel nucleo di LaTeX.

Bisogna però ricordare che fra l'editor e il programma di composizione non c'è un collegamento diretto, ma i due programmi comunicano solo attraverso il file `.tex`.

Ecco perché questo file dovrebbe poter trasferire l'informazione della codifica di ingresso che l'editor ha usato per scriverlo al programma di composizione affinché questo possa recuperare l'informazione da trasferire correttamente nel file di uscita sotto forma di documento composto tipograficamente.

D'altro canto sarebbe opportuno che quando un file sorgente viene generato o modificato mediante un editor usando una certa codifica, esso venisse salvato su disco con la stessa codifica. Sembra ovvio, ma non è così: dipende dalla qualità dell'editor.

Per esperienza chi scrive può affermare che egli conosce due editor TeXShop (specifico delle macchine Macintosh) e TeXworks (multipiattaforma) che accettano delle righe di configurazione che permettono di adattare le impostazioni del programma alle informazioni fornite con queste righe; se ne parlerà più avanti. C'è l'editor Aquamacs (solo per macchine Macintosh) che può fare più o meno le stesse cose, ma con righe di configurazione che seguono una sintassi diversa da quella dei programmi menzionati sopra; anche emacs (multipiattaforma) accetta le stesse righe di configurazione che usa Aquamacs. Esistono altresì gli editor TeXstudio e Texmaker che riescono a distinguere automaticamente se un file sorgente è stato creato o modificato con la codifica UTF-8 oppure con la codifica ISO Latin 1; in realtà sono in grado di distinguere i file codificati UTF-8 dagli altri, per cui un file che non sia UTF-8 viene classificato come ISO Latin 1. Questa capacità in generale è sufficiente, ma la codifica ISO Latin 1 non è la sola codifica dove le lettere accentate nazionali sono codificate con un solo byte ma, come si è visto nella tabella 26.2; molte codifiche hanno i caratteri nazionali codificati con un solo byte, quindi il riconoscimento ISO Latin 1 potrebbe essere sbagliato; non succede spesso perché la codifica *applemac* viene usata relativamente poco, ma questo poco è già troppo... Più avanti si vedrà come riconoscere quale sia la codifica di un file `.tex`. Con l'uscita nel 2012 della versione 2.4 di TeXstudio, la documentazione continua a dire quanto riportato sopra, ma di fatto TeXstudio è in grado di decifrare le righe magiche con la sintassi di TeXShop.

I dettagli sono molto tecnici e si omettono; non conosco nessun documento dove questi dettagli siano descritti in modo comprensibile; per quel che riguarda la codifica di uscita esiste il file `encguide.pdf`, leggibile con il solito coman-

Tabella 26.2: Alcune codifiche di ingresso per i caratteri latini

Codice	ansinew	latin1	latin9	applemac	cp437	cp1250
128	€			Ä	Ç	€
129				Å	ü	
130	‚			Ç	é	‚
131	*f*			É	â	
132	„			Ñ	ä	„
133	…			Ö	à	…
134	†			Ü	å	†
135	‡			á	ç	‡
136	ˆ			à	ê	
137	‰			â	ë	‰
138	Š			ä	è	Š
139	‹			ã	ï	‹
140	Œ			å	î	Ś
141				ç	ì	Ť
142	Ž			é	Ä	Ž
143				è	Å	Ź
144				ê	É	
145	‘			ë	æ	‘
146	’			í	Æ	’
147	“			ì	ô	“
148	”			î	ö	”
149	•			ï	ò	•
150	–			ñ	û	–
151	—			ó	ù	—
152	˜			ò	ÿ	
153	™			ô	Ö	™
154	š			ö	Ü	š
155	›			õ	¢	›
156	œ			ú	£	ś
157				ù	¥	ť
158	ž			û	Pts	
159	Ÿ			ü	*f*	ź
160				†	á	
161	¡	¡	¡	°	í	ˇ
162	¢	¢	¢	¢	ó	˘
163	£	£	£	£	ú	Ł
164	¤	¤	€	§	ñ	¤
165	¥	¥	¥	•	Ñ	Ą
166	¦	¦	Š	¶	ª	¦

Continua nella pagina seguente

Continua dalla pagina precedente

Codice	ansinew	latin1	latin9	applemac	cp437	cp1250
167	§	§	§	ß	º	§
168	¨	¨	š	®	¿	¨
169	©	©	©	©	¬	©
170	ª	ª	ª	™	½	Ş
171	«	«	«	´	¼	«
172	¬	¬	¬	¨	¡	¬
173				≠	«	
174	®	®	®	Æ	»	®
175	¯	¯	¯	Ø		Ż
176	°	°	°	∞		°
177	±	±	±	±		±
178	²	²	²	≤		˛
179	³	³	³	≥		ł
180	´	´	Ž	¥		´
181	µ	µ	µ	µ		µ
182	¶	¶	¶	∂		¶
183	·	·	·	Σ		·
184	¸	¸	ž	Π		¸
185	¹	¹	¹	π		ą
186	º	º	º	∫		ş
187	»	»	»	ª		»
188	¼	¼	Œ	º		Ľ
189	½	½	œ	Ω		˝
190	¾	¾	Ÿ	æ		ľ
191	¿	¿	¿	ø		ż
192	À	À	À	¿		Ŕ
193	Á	Á	Á	¡		Á
194	Â	Â	Â	¬		Â
195	Ã	Ã	Ã	√		Ă
196	Ä	Ä	Ä	*f*		Ä
197	Å	Å	Å	≈		Ĺ
198	Æ	Æ	Æ	Δ		Ć
199	Ç	Ç	Ç	«		Ç
200	È	È	È	»		Č
201	É	É	É	…		É
202	Ê	Ê	Ê			Ę
203	Ë	Ë	Ë	À		Ë
204	Ì	Ì	Ì	Ã		Ě
205	Í	Í	Í	Õ		Í
206	Î	Î	Î	Œ		Î

Continua nella pagina seguente

Continua dalla pagina precedente

Codice	ansinew	latin1	latin9	applemac	cp437	cp1250
207	Ï	Ï	Ï	œ		Ď
208	Đ	Đ	Đ	–		Đ
209	Ñ	Ñ	Ñ	—		Ń
210	Ò	Ò	Ò	“		Ň
211	Ó	Ó	Ó	”		Ó
212	Ô	Ô	Ô	‘		Ô
213	Õ	Õ	Õ	’		Ő
214	Ö	Ö	Ö	÷		Ö
215	×	×	×	◇		×
216	Ø	Ø	Ø	ÿ		Ř
217	Ù	Ù	Ù	Ÿ		Ů
218	Ú	Ú	Ú	/		Ú
219	Û	Û	Û	¤		Ű
220	Ü	Ü	Ü	‹		Ü
221	Ý	Ý	Ý	›		Ý
222	Þ	Þ	Þ	fi		Ţ
223	ß	ß	ß	fl		ß
224	à	à	à	‡	α	ŕ
225	á	á	á	·	β	á
226	â	â	â	,	Γ	â
227	ã	ã	ã	„	π	ă
228	ä	ä	ä	‰	Σ	ä
229	å	å	å	Â	σ	Ĺ
230	æ	æ	æ	Ê	μ	ć
231	ç	ç	ç	Á	γ	ç
232	è	è	è	Ë	Φ	č
233	é	é	é	È	θ	é
234	ê	ê	ê	Í	Ω	ę
235	ë	ë	ë	Î	δ	ë
236	ì	ì	ì	Ï	∞	ě
237	í	í	í	Ì	ϕ	í
238	î	î	î	Ó	ε	î
239	ï	ï	ï	Ô	$\cap$	ď
240	ð	ð	ð		$\equiv$	đ
241	ñ	ñ	ñ	Ò	$\pm$	ń
242	ò	ò	ò	Ú	$\geq$	ň
243	ó	ó	ó	Û	$\leq$	ó
244	ô	ô	ô	Ù		ô
245	õ	õ	õ	ı		ő

Continua nella pagina seguente

Continua dalla pagina precedente

Codice	ansinew	latin1	latin9	applemac	cp437	cp1250
246	ö	ö	ö	ˆ	÷	ö
247	÷	÷	÷	˜	≈	÷
248	ø	ø	ø	¯	°	ř
249	ù	ù	ù	˘	∙	ů
250	ú	ú	ú	˙	•	ú
251	û	û	û	˚	√	ű
252	ü	ü	ü	¸	^{n}	ü
253	ý	ý	ý	˝	^{2}	ý
254	þ	þ	þ	˛	■	ţ
255	ÿ	ÿ		ˇ		˙

do `texdoc encguide`, dove vengono descritte alcune delle codifiche latine, in particolare la codifica *T1*.

Però non è particolarmente difficile leggere i file, per esempio, `latin1.def` e `t1enc.def`[2]; il primo descrive la corrispondenza fra i caratteri della tastiera codificati secondo la norma ISO 8859-1, nota anche come ISO Latin 1, e la codifica interna di pdflatex. Il secondo descrive come si passi dalla codifica interna del sistema TEX alla codifica dei font usati per il testo composto, e quindi alla scelta dei glifi giusti da riportare nel file di uscita.

26.2.1 Situazioni di default

Per quel che riguarda pdflatex se un file da comporre non contiene nessuna informazione relativa alla codifica del file stesso dal 2018 viene assunta di default la codifica *utf8*. Questo vuol dire che se non si è specificata nessuna opzione né il pacchetto *inputenc*, pdflatex per impostazione predefinita si comporta come se nel preambolo fosse stata presente l'istruzione `\usepackage[utf8]{inputenc}`. Analogamente, se il file sorgente non contiene nessuna informazione relativa ai font da usare per comporre il file di uscita, per impostazione predefinita assume i caratteri Computer Modern con la codifica a 7 bit *ot1*.

Queste impostazioni predefinite possono andare bene per la maggior parte dele lingue occidentali; la composizione ha luogo senza particolari problemi, ma l'eventuale cesura in fn di riga potrebbe essere limitata nelle sue prestazioni per le lingue che fanno uso di segni diacritici. Per questo motivo dal 2018, scrivendo in lingue diverse dall'inglese, è opportuno non affidarsi alle impostazioni predefinite, almeno per la scelta dei font con cui comporre il file di uscita.

Invece, se si devono modificare e ricomporre file sorgente creati prima del 2018, è necessario avere tutte le accortezze descritte nei paragrafi che seguono.

[2]Si trovano entrambi nella cartella `.../tex/latex/base/`.

26.2.2 La codifica di ingresso

Se si apre il file `latin1.def`, si vede che le prime righe definiscono con `\ProvideTextCommandDefault` il significato di default di alcuni comandi interni generalmente conosciuti da parte dei programmi di composizione; per esempio, il primo comando

```
\ProvideTextCommandDefault{\textdegree}{\ensuremath{{^\circ}}}
```

dice che se `\textdegree` non è ancora stato definito, esso va composto di default in modo matematico con un circoletto ad esponente. Naturalmente esistono dei pacchetti per i font di uscita che possono definire o ridefinire questo comando di default per produrre un segno più adeguato, disegnato apposta, ma in mancanza di questi font la definizione di default produce un risultato accettabile.

Infatti `\textdegree`, tramite il pacchetto *textcomp*, produce il segno °, mentre il circoletto ad esponente è °, come si vede, produce un segno leggermente più grande ma è accettabilissimo.

Ci sono anche dei comandi del tipo:

```
\ProvideTextCommandDefault{\textcent}
   {\TextSymbolUnavailable\textcent}
```

che emettono un messaggio di errore se si usa il comando `\textcent`; questo succede perché il segno ¢ non è facilmente simulabile in un modo che vada bene per ogni font, codifica, famiglia, serie, forma e corpo; per produrlo ci vuole dunque l'uso di una polizza di caratteri in uscita che contenga questo segno e che perciò dia una definizione valida per `\textcent`.

Dopo altri comandi di questa specie, vengono definite le corrispondenze fra i caratteri con gli indirizzi da 161 a 255 nella codifica ISO 8859-1 e la codifica interna di pdflatex. Per esempio la definizione

```
\DeclareInputText{164}{\textcurrency}
```

specifica che introducendo con la tastiera nello *shell editor* il carattere che il driver della tastiera trasforma nel codice 164, questo produce sullo schermo il segno ¤. È vero che fra i primi comandi si dichiara che il segno corrispondente al comando `\textcurrency` non è disponibile, ma se questo comando è già stato definito o viene ridefinito in altro modo, questo segno potrebbe venire stampato nel modo giusto e non verrebbe emesso nessun avviso di simbolo non disponibile; infatti qualche riga più in alto il segno è stato stampato correttamente, perché la composizione di questo testo si affida anche alle funzionalità del pacchetto *textcomp*, già di default, che permettono di usare i segni della collezione Text Companion Font illustrati nella figura 18.10.

Per i caratteri in entrata non strettamente ASCII, come le lettere nazionali si hanno poi svariate definizioni del tipo seguente, valido per la lettera 'ï' che nella codifica di entrata ISO Latin 1 occupa la posizione 239:

```
\DeclareInputText{239}{\"\i}
```

Questa dichiarazione sostituisce il carattere 'ï' contenuto nel file di entrata con quanto è contenuto nel secondo argomento, ma del quale l'utente normale non deve preoccuparsi; esso serve per porre la dieresi sulla lettera 'ı' senza puntino indicato con la macro `\i`. Oggi l'utente normale non deve preoccuparsi della 'i senza puntino' (a meno che non scriva in irlandese o in turco) perché ci pensa sempre LaTeX a usarla quando occorre, almeno in modalità testo. Si vede dunque che anche in questo caso il carattere non appartenente al sottoinsieme ASCII viene sostituito con codici interni di LaTeX.

Si noti perciò che con la codifica di entrata (le altre disponibili con il sistema TeX sono concepite nello stesso modo ma, ovviamente, con definizioni diverse, come è stato precedentemente mostrato con alcuni esempi nella tabella 26.2) ogni tasto che produca nello *shell editor* un segno diverso dai caratteri ASCII viene trasformato in un segno visibile sullo schermo e nel file `.tex` che si sta scrivendo o modificando. Quando il file `.tex` verrà letto da pdflatex ogni carattere che vi si trova viene sostituito da comandi interni di LaTeX e verrà trattato come tale, finché non verrà il momento di trasferire il testo composto nel buffer di uscita costituito dalla famosa scatola 255 riservata da TeX a questo scopo.

Si ricorda che i primi 128 caratteri di ogni codifica di ingresso corrispondono sempre ai caratteri definiti dall'*American Standard Code for Information Interchange* – ASCII, che comprendono i 32 codici per il funzionamento delle telescriventi[3], le cifre, i segni di interpunzione e l'alfabeto latino di 26 lettere non accentate, in forma maiuscola e minuscola, per un totale di $128 - 32 = 96$ segni stampabili (in realtà il centoventottesimo non è stampabile e quindi ne restano solo 95). Da una codifica ISO all'altra cambiano i secondi 128 caratteri, come si è visto con la tabella 26.2, ed è per questo che esistono diversi nomi per distinguere le varie codifiche di ingresso. Ognuno dei file `.def` relativo alla codifica di ingresso definisce i comandi testuali di default e la corrispondenza fra i codici da 128 a 255 con i comandi interni di LaTeX (in alcune codifiche gli indirizzi da 128 a 160 non ricevono nessuna definizione).

Attenzione: normalmente ogni *shell editor* salva i file con la codifica predefinita nella sua configurazione. TeXShop e TeXworks possono fare di più: possono salvare un file scritto con una certa codifica anche con una codifica diversa da quella corrispondente alla sua configurazione. Di solito questa funzionalità non è un problema, anzi può essere molto utile per cambiare codifica ad un file. Bisogna solo stare attenti a non confondersi sbagliando preferenze nel pannello di dialogo per salvare o esportare un file, oppure nella apposita finestrella in basso a destra

[3]Cose del passato, ma non completamente. Infatti contengono anche i codici di fine riga e di tabulazione che il sistema TeX sa trattare in modo particolare; i codici di tabulazione per l'editor serve per far spostare il cursore fino alla successiva posizione di tabulazione, ma per pdflatex quel codice viene trattato come un semplice spazio. Invece i codici di fine riga `<LF>` e `<CR>` sono usati in modi diversi dai varii sistemi operativi e, quindi, dagli editor che lavorano su macchine con sistemi operativi diversi. Anche questo è un tormentone, come le codifiche, ma i programmi del sistema TeX sono già predisposti per evitare all'utente di doversene preoccupare. I vari editor invece possono avere qualche problemino quando viene aperto un file composto con un codice di fine riga diverso da quello che l'editor si aspetta.

nella cornice di TeXworks; non è comune fare questi errori, ma vale la pena segnalare la necessità di stare attenti.

26.2.3 La codifica di uscita

La codifica dei font latini *T1* nelle prime 32 posizioni contiene glifi di servizio che comprendono gli accenti ed altri segni speciali; i successivi 95 caratteri sono i caratteri ASCII; dalla posizione 127 alla posizione 255 la polizza di caratteri contiene quasi tutti i segni necessari per scrivere in tutte le lingue dell'Europa occidentale; per l'Europa orientale alcuni segni vengono creati per sovrapposizione dell'accento alla lettera base, ma sono pochissimi i segni che richiedono questo trattamento; si veda per esempio la tabella 18.5.

Il file `t1enc.def` contiene le definizioni delle corrispondenze fra i comandi interni di LaTeX e le posizioni dei singoli segni nella tabella del font di uscita. La sintassi è un po' diversa da quella usata per la codifica di ingresso, ma il significato è altrettanto comprensibile. Per esempio:

```
\DeclareTextAccent{\‘}{T1}{0}
\DeclareTextAccent{\’}{T1}{1}
```

definiscono gli accenti grave e acuto mettendo in relazione i comandi LaTeX `\‘` e `\’` con la posizione dei corrispondenti segni (0 e 1, rispettivamente) nella polizza conforme alla codifica *T1*. Di fatto questi accenti possono servire solo per essere sovrapposti a lettere che non hanno corrispondenza nella codifica del font di uscita *T1*, per esempio: s̀.

```
\DeclareTextCommand{\c}{T1}[1]
   {\leavevmode\setbox\z@\hbox{#1}\ifdim\ht\z@=1ex\accent11 #1%
     \else{\ooalign{\unhbox\z@\crcr
        \hidewidth\char11\hidewidth}}\fi}
```

ma si dice che il comando per la cediglia `\c` trova il segno 'cediglia' nella posizione 11 per metterlo sotto la lettera che costituisce il suo unico argomento. Però per i segni con la cediglia presenti nella polizza codificata con la codifica *T1*, come ç, ţ, ş,[4] ulteriori comandi dicono che cosa fare per trovare il segno giusto nella polizza dei font.

Successivamente vengono definite le corrispondenze fra i simboli speciali e/o accentati e gli indirizzi in cui trovare questi segni nella polizza codificata *T1*. Per esempio:

```
\DeclareTextSymbol{\ss}{T1}{255}
...
\DeclareTextComposite{\"}{T1}{i}{239}
\DeclareTextComposite{\"}{T1}{\i}{239}
```

[4] Questi ultimi due, se usati in rumeno, non dovrebbero avere la cediglia, ma una virgola; spesso però i rumeni devono accontentarsi di una approssimazione, perché molti font mancano del segno con la virgola; esiste il pacchetto *combelow* per risolvere con LaTeX questo problema.

La prima dichiarazione serve per indicare che il segno ß ottenibile con il comando interno `\ss` si trova nella posizione 255 della polizza. Invece il segno ottenuto usando il comando per la dieresi con l'argomento `i` (i con il puntino) oppure `\i` (ı senza puntino) producono entrambi la 'ï' già accentata[5] che si trova nella posizione 239 della polizza dei font codificati *T1*.

Se invece non si è specificata nessuna opzione per la codifica del font di uscita, LaTeX usa la codifica di default *OT1*, tabella 18.3; per questa codifica aprendo il file `ot1enc.def` si trovano le seguenti dichiarazioni:

```
\DeclareTextAccent{\"}{OT1}{127}
\DeclareTextSymbol{\i}{OT1}{16}
\DeclareTextCompositeCommand{\"}{OT1}{i}{\"\i}
```

La prima dichiarazione indica dove trovare nella polizza dei caratteri il segno della dieresi; la seconda dice dove trovare nella polizza dei caratteri la 'i' senza puntino; la terza dice che se in entrata si è scritto `\"i` invece che direttamente `ï`, la sequenza per il comando della dieresi seguita dalla lettera i col puntino deve essere interpretata alla stessa maniera di come si interpreta il codice di ingresso `ï`, cioè come `\"\i`. Come si vede, entrambe le dichiarazioni sono valide solo per la codifica *OT1* dei font di uscita.

Quando quindi LaTeX incontra una 'i' accentata o con la macro per l'accento o rappresentata direttamente con il codice d'entrata per la 'i' accentata, si ritrova a dover eseguire l'espressione `\"\i`; questa si traduce nei comandi primitivi che formano la stringa `\accent127\char16`; il comando primitivo `\accent` serve per sovrapporre il segno che ha l'indirizzo specificato come suo argomento sul segno seguente che qui è indicato con il comando primitivo `\char` seguito dall'indirizzo specifico della 'i' senza puntino. Quei comandi primitivi verranno eseguiti solo quando il capoverso sarà stato elaborato sotto forma di righe giustificate con le eventuali cesure in fin di riga, pronto per essere conservato nella pagina in costruzione; ma quando l'algoritmo di divisione in righe entra in gioco, agisce sul testo che contiene ancora questi comandi primitivi. Come tali non fanno parte della stringa di token leciti in una parola, per cui l'algoritmo di sillabazione termina la 'parola' subito prima del comando per l'accento, cosicché non riesce a sillabare completamente l'intera stringa che gli umani considerano una parola.

Ora dovrebbe quindi essere chiaro che:

1. Quando pdflatex deve eseguire la sillabazione usando per l'uscita la codifica *T1* si trova nel suo buffer interno il capoverso da dividere in righe composto solo da caratteri, e non contiene comandi di nessun genere (tranne quelli che non riguardano il testo); e quindi funziona a dovere. Invece con la codifica *OT1*, quella di default, ogni comando per inserire gli accenti resta nel buffer del capoverso e abbiamo visto che questo impedisce a pdflatex

[5] Questo fatto succede con quasi tutti gli accenti quando l'argomento è una 'i'. Per cui diventa obsoleta la prescrizione che per mettere un accento sopra la 'i' bisogna usare *solo* la variante senza puntino.

di riconoscere come parole le stesse stringhe di caratteri che gli umani considerano come tali.

2. I segni accentati delle polizze codificate *T1* sono disegnati individualmente quindi lo stesso accento è collocato sulla lettera di base nella posizione più opportuna in relazione alla sua forma. Invece con la codifica *OT1* lo stesso accento viene sempre centrato sulla mezzeria della lettera di base senza tenere conto delle asimmetrie ottiche associate a ciascuna lettera di base.

Queste sono due ottime ragioni per non usare mai i font con codifiche diverse da *T1* quando si compongono con pdflatex testi in lingue che si scrivono in caratteri latini e che contengono diversi diacritici; in realtà questa raccomandazione vale anche per l'inglese, che solitamente non usa diacritici, ma li usa (*a*) nelle citazioni di nomi o parole straniere, e (*b*) nelle edizioni storiche o critiche di testi scritti in *middle English*. Si usi sempre, comunque, la codifica *utf8* per la codifica d'entrata.

Discorsi simili si possono fare per la codifica *utf8*, solo che ogni carattere UNICODE che si introduce con la tastiera attraverso lo *shell editor* nel file `.tex` non è formato da un solo byte, ma da un codice formato spesso da diversi byte, per cui la gestione dell'immissione dei caratteri nel file `.tex` è un poco più complicata; per fortuna ci pensa l'editor e l'utente è sollevato dalla necessità di occuparsi di questi dettagli. Anche la trasformazione dei caratteri immessi con eventuali comandi TEX o LATEX diventa un poco più complessa, ma ci pensa il programma di composizione. L'uso della codifica *utf8* offre però il vantaggio che, tastiera permettendo, qualunque segno immesso con la tastiera può essere fatto corrispondere al segno giusto, compatibilmente con la codifica dei font di uscita, cioè con l'effettiva presenza di quel segno nella polizza dei font di uscita. Se anche questi font sono codificati UNICODE, non c'è bisogno di nessun passaggio intermedio, ed è questo quello che fanno gli interpreti xetex, luatex e context-mk-iv.

26.3 Specificare la codifica giusta

Qualsiasi sia la codifica dei caratteri di ingresso, nasce un problema che ogni utente di LATEX deve affrontare.

Egli deve specificare (mediante l'uso dell'opzione giusta al pacchetto *inputenc* nel preambolo del suo documento) quale sia la codifica con il quale ha scritto o modificato il file sorgente, altrimenti il suo file non potrà essere correttamente compilato. Questa specificazione non serve, se usa i programmi lualatex o xelatex quando usa solo i font OpenType.

Un errore comune è quello di invocare il pacchetto ma di specificare una opzione non coerente con le impostazioni dello *shell editor*. Succede più spesso di quanto non si creda. Tizio scrive un file sorgente `.tex` usando la codifica di ingresso *utf8*; poi spedisce il file al suo collega Caio. Caio, il cui *shell editor* è configurato per usare la codifica *latin1*, apre il file inviatogli da Tizio, ma trova il testo intercalato con caratteri strani. Questa è un situazione frequentissima.

È successo anche a chi scrive (utente di calcolatori Mac) di ricevere da un'altra persona, anch'essa utente di un Mac, un file LaTeX codificato con la codifica MacRoman (*applemac*); l'altra persona non si rendeva conto che quella è una codifica ristretta ai sistemi Mac e non è diffusa come le codifiche 'apolidi', non legate ad un particolare sistema operativo o a una marca di hardware. Chi scrive abitualmente usa la codifica UTF-8 sia per scrivere in italiano sia per scrivere con altri alfabeti; quando ha ricevuto il file ci ha messo un poco a rendersi conto di questa situazione e quando se ne è accorto era troppo tardi per porvi rimedio in modo corretto. Quanto segue, oltre a descrivere la situazione relativa ai malintesi generati dalle varie codifiche, serve anche per indicare alcuni modi corretti per affrontare questo tipo di problemi.

Se Tizio e Caio stanno usando lo *shell editor* multipiattaforma TeXworks e se hanno configurato bene il file `.tex` non devono fare nulla. Per configurare bene il file bisogna che esso, prima del testo vero e proprio, contenga delle righe di configurazione costituite da righe di commento, scritte in modo speciale, che dicono a TeXworks se esiste e come si chiama il master file e quale sia la codifica usata per scrivere il testo (nell'esempio che segue la prima riga è facoltativa, nel senso che se il file è il master file, non c'è nessun bisogno di fornire questa informazione lapalissiana)[6]; inoltre la scelta di Isolatin è solo un esempio, perché sarebbe meglio orientarsi sempre sulla codifica UTF-8:

```
%␣!TeX␣root␣=␣MyMasterFile.tex
%␣!TeX␣encoding␣=␣IsoLatin
%␣!Tex␣TS-program␣=␣pdflatex
```

In questo modo, se Tizio e Caio usano *shell editor* diversi, ma che siano in grado di decifrare queste speciali righe di commento, sono gli editor che si autoconfigurano di conseguenza e i due tastieristi non devono fare assolutamente nulla.

Esistono poi alcuni *shell editor* che riconoscono automaticamente la codifica usata per creare i file. Apparentemente Aquamacs è uno di questi *shell editor*; ho provato a far leggere ad Aquamacs un file salvato con la codifica *applemac* (MacRoman) e sullo schermo è apparso il file con le lettere accentate sostituiti dai loro codici ottali preceduti dal backslash; quindi a stretto rigore Aquamacs riconosce automaticamente molte codifiche, ma non riconosce "Western MacRoman" sebbene Aquamacs sia un programma specifico per le macchine Macintosh[7]; tuttavia la sostituzione globale di quei codici numerici ottali con le corrispondenti lettere accentate è molto facile.

L'editor Aquamacs dispone di altre righe speciali che seguono una sintassi diversa:

```
% -*- mode: latex; coding: latin-1;  TeX-master: MyMasterFile.tex -*-
```

[6] Le righe indicate qui di seguito seguono la sintassi di TeXShop; TeXworks ha un'altra sintassi, ma capisce anche la sintassi di TeXShop.

[7] In effetti dando ad Aquamacs il comando per esporre le codifiche che può riconoscere, la codifica Western MacRoman non è elencata.

Questa riga speciale deve essere all'inizio del file `.tex`; i due insiemi di righe speciali, quelle per Aquamacs (valide anche per emacs) e quelle per TeXShop/ TeXworks, possono coesistere nello stesso file in questo modo:

```
% -*- mode: latex; coding: latin-1;  TeX-master: MyMasterFile.tex -*-
% !TeX root = MyMasterFile.tex
% !TeX encoding = IsoLatin
% !Tex TS-program = pdflatex
```

e gli editor che le possono interpretare lo fanno senza confondersi con le righe presenti per editor diversi.

L'editor TeXstudio nelle versioni precedenti alla 2.4 è in grado di distinguere la codifica *utf8*, e classifica come ISO Latin 1 ogni altra codifica; il che può essere errato e bisogna intervenire a mano. A partire dalla versione 2.4 TeXstudio è in grado di interpretare le righe magiche e anche di convertire la codifica a scelta dell'utente.

Per impostare le righe speciali con TeXworks bisogna procedere a mano. Con TeXShop è più semplice perché il menù `Macros` contiene esplicitamente le voci `Encoding`, `Program` e `Root`; scegliendo ciascuna di queste voci si aprono pannelli di dialogo fra cui scegliere l'informazione giusta. Per Aquamacs alcune "variabili" della riga speciale devono essere introdotte a mano, ma per il nome del master file si può scegliere dal menù `LaTeX/Multifile/Parsing/Set Master File` che apre un pannello di dialogo dove scegliere il nome del file che deve svolgere il ruolo di master file.

26.3.1 Scoprire la codifica di input

Se Tizio e Caio non si trovano in queste condizioni, che cosa deve fare Caio per poter elaborare il testo contenuto nel file ricevuto da Tizio? Ha due possibilità: (*a*) cambia codifica al file `.tex`, oppure (*b*) cambia l'impostazione della codifica del suo *shell editor*. Ma prima di tutto deve scoprire quale sia la codifica del file ricevuto da Tizio.

La soluzione (*a*) sarebbe preferibile; e sarebbe preferibile anche che entrambi usassero sempre la codifica *utf8*. Se Tizio poteva compilare correttamente il suo file, era perché aveva specificato l'opzione giusta al pacchetto *`inputenc`*; Caio, allora, benché non riesca a leggere il testo del documento sorgente con facilità, può però leggere perfettamente il preambolo; basta allora che cerchi la stringa `inputenc` e lì trova che Tizio ha usato la codifica *latin1*. Chiude il file appena aperto, *ma senza salvarlo*, telefona a Tizio ricordandogli che avevano convenuto di usare entrambi la codifica *utf8*, riapre il file con TeXworks e procede come spiegato nel paragrafo 26.5 salvando il file con la codifica *utf8*.

Fin qui va bene. Ma se Tizio ha usato la codifica *latin1* e si è scordato di specificare la codifica del suo *shell editor* nel preambolo del documento? Be', vuol dire due cose: primo, non poteva compilare il suo file producendo il file di uscita in modo corretto e, secondo, Caio deve tirare a indovinare per scoprire la codifica usata da Tizio, probabilmente in modo inconsapevole; è chiaro che

basterebbe una telefonata o un messaggio di posta elettronica; è altrettanto chiaro che il procedimento indicato nel paragrafo 26.5 permette di scoprire sia la codifica usata sia di eseguire la conversione. Ma talvolta il file è stato scritto da un anonimo Tizio che l'ha messo in rete e Caio l'ha trovato interessante guardandolo sullo schermo; ma dopo averlo scaricato non riesce più a leggerlo. In questo caso disperato, sempre senza modificare e salvare il file appena scaricato, tira ad indovinare; l'indovinello è facile se il file era stato scritto usando la codifica *utf8*, perché ogni carattere non ASCII è sostituito da due o più caratteri strani. Invece l'indovinello può essere difficile se appaiono caratteri strani isolati, specialmente alla fine delle parole. Non resta che seguire la procedura indicata nel paragrafo 26.5 e non ci sono più problemi.

Il caso più disperato si ha quando Tizio e/o Caio si servono di *shell editor* vecchiotti, che usano una sola codifica, ma purtroppo si tratta di codifiche diverse. Non c'è quindi modo di modificare le impostazioni dello *shell editor* e non rimane altra scelta che quella di cambiare, mediante appositi programmi da linea di comando, la codifica del file `.tex`. O meglio, se i due collaboratori sanno di essere in queste condizioni avranno cura di scrivere o modificare i file `.tex` sorgente del loro lavoro cooperativo usando solamente caratteri ASCII; in sostanza non useranno direttamente i caratteri nazionali della loro tastiera, ma scriveranno le varie sequenze per gli accenti come richiesto dalla sintassi originale di TEX: per l'italiano useranno perciò `\‘a`, `\‘e`, `\’e`, `\‘i`, `\‘o`, `\‘u`, e si comporteranno in modo simile con le lettere maiuscole. È chiaro che questo modo di procedere, formalmente corretto, non è molto efficiente e i due collaboratori potrebbero accordarsi per aggiornare il loro sistemi TEX e per usare lo stesso fra uno dei vari *shell editor* multipiattaforma. In ogni caso seguendo la procedura indicata nel paragrafo 26.5 possono entrambi convertirsi reciprocamente i file nel modo che vogliono. In fondo TeXworks è multipiattaforma e non è legato a nessuna particolare distribuzione del sistema TEX. Quindi se TeXworks non fosse già installato, basta che entrambi i collaboratori lo installino. Possono anche usarlo per lavorare sui loro file, non solo per scoprire la codifica oppure per convertirne la codifica; è un editor semplice, ma ha diverse caratteristiche molto interessanti.

Si tenga presente che il modo apparentemente primitivo di scrivere i caratteri accentati secondo la sintassi TEX, consente molte più combinazioni di quelle che consentirebbe la codifica UNICODE, perché ogni segno diacritico di cui TEX dispone (una dozzina) può essere messo su qualunque lettera latina (26) minuscola o maiuscola, con un totale di combinazioni enorme, che consentono anche delle combinazioni "impossibili", cioè segni che non vengono usati in nessuna lingua; questo meccanismo non è affatto tanto primitivo come si potrebbe pensare, tant'è che costituisce l'ossatura della codifica interna del sistema TEX, indipendentemente dalla codifica di ingresso e da quella relativa ai font di uscita.

26.3.2 Cambiamento della codifica

Ma se proprio si deve cambiare codifica e, per qualche motivo, non si volesse installare TeXworks, tra i programmi disponibili, quello base è l'applicativo iconv

per macchine Linux e Mac OS X, la cui sintassi è:

```
iconv [-c] [-s] [-f ⟨codifica1⟩] [-t ⟨codifica2⟩] ⟨inputfile⟩
```

dove le opzioni `-c` e `-s` controllano i messaggi d'errore nel caso che il file di ingresso contenga caratteri non convertibili. Il file di ingresso ⟨*inputfile*⟩ è scritto con la ⟨*codifica1*⟩ specificata con l'opzione `-f`, mentre il file di uscita è scritto con la ⟨*codifica2*⟩ specificata con l'opzione `-t`. Il programma di default scrive il file di uscita nello stream di scrittura standard (generalmente lo schermo); se si vuole scrivere su un file vero e proprio bisogna reindirizzare l'uscita verso un file tramite il codice di reindirizzamento:

```
iconv [-c] [-s] [-f ⟨codifica1⟩] [-t ⟨codifica2⟩] ⟨inputfile⟩ > ⟨outputfile⟩
```

dove il nome del file di uscita, ⟨*outputfile*⟩, deve essere diverso dal nome del file di entrata.

Per conoscere le codifiche disponibili sulla particolare macchina sulla quale si sta lavorando e per scrivere i nomi delle codifiche con l'ortografia giusta, conviene dare il comando:

```
iconv -l
```

e sul terminale appare tutto l'elenco delle codifiche disponibili per le quali è possibile eseguire le conversioni. Le codifiche scritte sulla stessa riga, stando alla pagina del manuale, sono sinonimi l'una dell'altra. Nella macchina sulla quale sto lavorando le codifiche disponibili (con le righe ripiegate e rientrate per i limiti della giustezza del testo) sono:

```
ANSI_X3.4-1968 ANSI_X3.4-1986 \textsc{ascii} CP367 IBM367 ISO-IR-6
    ISO646-US ISO_646.IRV:1991 US US-\textsc{ascii} CS\textsc{ascii}
UTF-8
ISO-10646-UCS-2 UCS-2 CSUNICODE
UCS-2BE UNICODE-1-1 UNICODEBIG CSUNICODE11
UCS-2LE UNICODELITTLE
ISO-10646-UCS-4 UCS-4 CSUCS4
UCS-4BE
UCS-4LE
UTF-16
UTF-16BE
UTF-16LE
UTF-32
UTF-32BE
UTF-32LE
UNICODE-1-1-UTF-7 UTF-7 CSUNICODE11UTF7
UCS-2-INTERNAL
UCS-2-SWAPPED
```

```
UCS-4-INTERNAL
UCS-4-SWAPPED
C99
JAVA
CP819 IBM819 ISO-8859-1 ISO-IR-100 ISO8859-1 ISO_8859-1
    ISO_8859-1:1987 L1 LATIN1 CSISOLATIN1
ISO-8859-2 ISO-IR-101 ISO8859-2 ISO_8859-2 ISO_8859-2:1987
    L2 LATIN2 CSISOLATIN2
ISO-8859-3 ISO-IR-109 ISO8859-3 ISO_8859-3 ISO_8859-3:1988
    L3 LATIN3 CSISOLATIN3
ISO-8859-4 ISO-IR-110 ISO8859-4 ISO_8859-4 ISO_8859-4:1988
    L4 LATIN4 CSISOLATIN4
CYRILLIC ISO-8859-5 ISO-IR-144 ISO8859-5 ISO_8859-5
    ISO_8859-5:1988 CSISOLATINCYRILLIC
ARABIC ASMO-708 ECMA-114 ISO-8859-6 ISO-IR-127 ISO8859-6
    ISO_8859-6 ISO_8859-6:1987 CSISOLATINARABIC
ECMA-118 ELOT_928 GREEK GREEK8 ISO-8859-7 ISO-IR-126 ISO8859-7
    ISO_8859-7 ISO_8859-7:1987 CSISOLATINGREEK
HEBREW ISO-8859-8 ISO-IR-138 ISO8859-8 ISO_8859-8 ISO_8859-8:1988
    CSISOLATINHEBREW
ISO-8859-9 ISO-IR-148 ISO8859-9 ISO_8859-9 ISO_8859-9:1989
    L5 LATIN5 CSISOLATIN5
ISO-8859-10 ISO-IR-157 ISO8859-10 ISO_8859-10 ISO_8859-10:1992
    L6 LATIN6 CSISOLATIN6
ISO-8859-13 ISO-IR-179 ISO8859-13 ISO_8859-13 L7 LATIN7
ISO-8859-14 ISO-CELTIC ISO-IR-199 ISO8859-14 ISO_8859-14
    ISO_8859-14:1998 L8 LATIN8
ISO-8859-15 ISO-IR-203 ISO8859-15 ISO_8859-15 ISO_8859-15:1998
ISO-8859-16 ISO-IR-226 ISO8859-16 ISO_8859-16 ISO_8859-16:2000
KOI8-R CSKOI8R
KOI8-U
KOI8-RU
CP1250 MS-EE WINDOWS-1250
CP1251 MS-CYRL WINDOWS-1251
CP1252 MS-ANSI WINDOWS-1252
CP1253 MS-GREEK WINDOWS-1253
CP1254 MS-TURK WINDOWS-1254
CP1255 MS-HEBR WINDOWS-1255
CP1256 MS-ARAB WINDOWS-1256
CP1257 WINBALTRIM WINDOWS-1257
CP1258 WINDOWS-1258
850 CP850 IBM850 CSPC850MULTILINGUAL
862 CP862 IBM862 CSPC862LATINHEBREW
866 CP866 IBM866 CSIBM866
MAC MACINTOSH MACROMAN CSMACINTOSH
```

```
MACCENTRALEUROPE
MACICELAND
MACCROATIAN
MACROMANIA
MACCYRILLIC
MACUKRAINE
MACGREEK
MACTURKISH
MACHEBREW
MACARABIC
MACTHAI
HP-ROMAN8 R8 ROMAN8 CSHPROMAN8
NEXTSTEP
ARMSCII-8
GEORGIAN-ACADEMY
GEORGIAN-PS
KOI8-T
MULELAO-1
CP1133 IBM-CP1133
ISO-IR-166 TIS-620 TIS620 TIS620-0 TIS620.2529-1 TIS620.2533-0
    TIS620.2533-1
CP874 WINDOWS-874
VISCII VISCII1.1-1 CSVISCII
TCVN TCVN-5712 TCVN5712-1 TCVN5712-1:1993
ISO-IR-14 ISO646-JP JIS_C6220-1969-RO JP CSISO14JISC6220RO
JISX0201-1976 JIS_X0201 X0201 CSHALFWIDTHKATAKANA
ISO-IR-87 JIS0208 JIS_C6226-1983 JIS_X0208 JIS_X0208-1983
    JIS_X0208-1990 X0208 CSISO87JISX0208
ISO-IR-159 JIS_X0212 JIS_X0212-1990 JIS_X0212.1990-0 X0212
    CSISO159JISX02121990
CN GB_1988-80 ISO-IR-57 ISO646-CN CSISO57GB1988
CHINESE GB_2312-80 ISO-IR-58 CSISO58GB231280
CN-GB-ISOIR165 ISO-IR-165
ISO-IR-149 KOREAN KSC_5601 KS_C_5601-1987 KS_C_5601-1989
    CSKSC56011987
EUC-JP EUCJP EXTENDED_UNIX_CODE_PACKED_FORMAT_FOR_JAPANESE
    CSEUCPKDFMTJAPANESE
MS_KANJI SHIFT-JIS SHIFT_JIS SJIS CSSHIFTJIS
CP932
ISO-2022-JP CSISO2022JP
ISO-2022-JP-1
ISO-2022-JP-2 CSISO2022JP2
CN-GB EUC-CN EUCCN GB2312 CSGB2312
CP936 GBK
GB18030
```

```
ISO-2022-CN CSISO2022CN
ISO-2022-CN-EXT
HZ HZ-GB-2312
EUC-TW EUCTW CSEUCTW
BIG-5 BIG-FIVE BIG5 BIGFIVE CN-BIG5 CSBIG5
CP950
BIG5-HKSCS BIG5HKSCS
EUC-KR EUCKR CSEUCKR
CP949 UHC
CP1361 JOHAB
ISO-2022-KR CSISO2022KR
```

Perciò per trasformare il file `myfile.tex` scritto con la codifica *latin1* in un altro file `myfile-utf8.tex` scritto con la codifica *utf8* bisogna dare il comando:

```
iconv -f LATIN1 -t UTF-8 myfile.tex > myfile-utf8.tex
```

Per convertire un file `myfilew.tex` scritto su una macchina Windows con la *code page* di default `1252` in un file con la codifica *utf8* si scriverà:

```
iconv -f WINDOWS-1252 -t UTF-8 myfilew.tex > myfile-utf8.tex
```

Per il Mac esiste un interfaccia grafica charco che permette di fare sostanzialmente le stesse cose lavorando prevalentemente con il mouse.

Per le macchine Windows si può usare ancora iconv ma bisogna installarlo dentro l'ambiente di simulazione CygWin operando come se si stesse installando il programma su una macchina Linux.

Alternativamente si può scaricare la libreria `libiconv` dal sito `http://sourceforge.net/projects/gnuwin32/files/libiconv/` e il programma di installazione provvede anche a rendere disponibile la versione Windows Winiconv con la quale si può operare senza ricorrere all'ambiente di simulazione CygWin.

Se tutto ciò dovesse 'disturbare' o comunque apparire troppo complicato, si riesamini la possibilità di sfruttare le capacità dello *shell editor* TeXworks, oppure si ricorra alla sintassi TEX che non fa uso di caratteri speciali ma solo caratteri ASCII e usa macro particolari per indicare quali segni diacritici usare.

26.4 Collage di contributi diversi

Un altro problema sorge invece quando si compone un testo eseguendo un collage di contributi di varie persone che hanno fornito file in formati come `.doc` o `.pdf`, e il compositore procede a mano copiando dal file esterno e incollando nel file `.tex`.

In questi casi spesso sono i caratteri di interpunzione quelli che differiscono sensibilmente dai caratteri della codifica che si vorrebbe usare, fosse essa anche

la codifica più generale *utf8*. A occhio nudo spesso questi caratteri (apostrofo, virgolette semplici, virgolette diritte, eccetera) sono difficili da rilevare perché il riconoscimento corretto dipende anche dal disegno dei font che vengono usati dallo *shell editor* nella finestra del file sorgente; io preferisco i font a spaziatura fissa in questa finestra del file sorgente ma, nonostante questo, quei segni sono difficili da rilevare ad occhio nudo. Tenendo conto che un file di ingresso deve essere scritto tutto con la stessa codifica, bisogna per forza convertire a mano tutti i segni nella codifica principale, oppure marcare il file sorgente con la codifica specifica usata dal collaboratore esterno o che il suo programma di trasformazione nel file `.pdf` ha usato.

TeXworks e TeXShoppotrebbero interpretare correttamente questa informazione e Aquamacs potrebbe diagnosticare da solo la codifica usata. In ogni caso si può eseguire la compilazione con pdflatex e poi andare a controllare un certo numero di istanze di questi caratteri speciali passando dalla finestra del file sorgente alla finestra del file composto sfruttando la capacità dello *shell editor* e del visualizzatore del file composto di eseguire correttamente la ricerca diretta e inversa. Se questi caratteri un po' particolari sono tutti presenti in modo corretto nel file di uscita, allora la codifica dichiarata o determinata è corretta. Altrimenti è opportuno procedere ad una correzione manuale di tutti questi segni; solitamente la ricerca e sostituzione cumulativa di *tutte* le istanze di un dato carattere può essere fatta con pochi click di mouse con qualunque *shell editor*.

26.5 Considerazioni riassuntive

In conclusione conviene fare le seguenti considerazioni che riassumono operativamente quanto si è detto nei paragrafi precedenti.

1. Se si usano i programmi lualatex o xelatex solamente con font OpenType, bisogna che lo *shell editor* sia capace di gestire in entrata i font codificati *utf8*, ma non bisogna né usare *inputenc* e *fontenc* né specificar loro alcuna opzione. Se si usano alcuni font Type 1 bisogna specificare la codifica di questi font prima di richiamare il file di estensione *fontspec* e bisogna definire opportune macro per comporre con quei font Type 1. Ciò è quanto è stato descritto nel paragrafo 15.7.2 nella pagina 372. Altrimenti se questi due programmi servono per comporre un file predisposto per essere composto con pdflatex, il file deve essere codificato e gestito completamente come per usare quest'ultimo programma di composizione e quindi seguendo i punti che seguono. — Per lualatex e xelatex
2. La codifica di ingresso deve essere sempre specificata come opzione nell'invocazione del pacchetto *inputenc*; questa codifica riguarda essenzialmente l'interazione fra la tastiera e lo *shell editor* e come questo salva il file sorgente sul disco. Ci si ricordi, tuttavia, che del 2018 la codifica UTF-8 è la codifica preimpostata per tutti i programmi di composizione del sistema — Solo per pdflatex

TEX; pdflatex conserva la facoltà di gestire anche le precedenti codifiche, se non altro per ricompilare documenti nati prima del 2018.

3. Per la perfetta composizione del documento finale bisogna sempre usare font il più possibile completi, comunque almeno con la codifica *T1* per migliorare la cesura in fin di riga e per avere i segni accentati di fattura migliore rispetto a quelli che si potrebbero avere sovrapponendo l'accento alla lettera di base. L'ideale sarebbe quello di usare i font OpenType, usabili solo con lualatex e xelatex che richiedono sia per l'editor sia per la composizione solo la codifica UNICODE con la sua transcodifica UTF-8.
4. È fortemente consigliabile scrivere le righe di autoconfigurazione per lo *shell editor*; questo vale anche se non si usa uno *shell editor* che le capisca e possa autoconfigurasi di conseguenza; esse rappresentano un buon promemoria permanente e sono utilissime per le altre persone a cui eventualmente si debba inviare il file sorgente.
5. Se si deve cambiare codifica, allora conviene procedere come segue.

   (a) È necessario eseguire subito una copia del file di cui si deve cambiare la codifica, ricordandosi di cambiare nome alla copia e di salvarlo con lo stesso nome in una cartella diversa; nel caso ci si sbagli con le operazioni successive, il file originale può sempre essere recuperato.
   (b) Se aprendo un file `.tex` con lo *shell editor* si vede che il file contiene dei caratteri strani, allora l'impostazione della codifica dello *shell editor* non corrisponde alla codifica del font con cui è stato scritto il file `.tex`. In questo caso si eviti di fare qualunque modifica al file e ci si guardi bene dal "salvarlo" su disco.
   (c) Si riapra il file con TeXworks; questo editor è disponibile per tutte le piattaforme e solo con le installazioni Linux potrebbe essere necessario installarlo espressamente. Per le operazioni che seguono è essenziale usare TeXworks, poi volendo si può tornare a qualunque altro editor che l'utente preferisca rispetto a TeXworks.
   (d) Una volta aperto il file che contiene caratteri strani, ci si porti su una pagina in cui se ne vedono alcuni, sostituiti, generalmente, da rombi neri contenenti all'interno un punto interrogativo bianco; si mantenga quella pagina e si prenda nota dei numeri di riga dove compaiono i rombi neri.
   (e) Nella riga inferiore della schermata di TeXworks ci sono in basso a destra delle finestrine, nella seconda delle quali c'è la codifica impostata di default per questo editor; cliccando su questa finestra si apre il lungo elenco di codifiche che TeXworks è capace di gestire; si scelga un'altra codifica: per esempio, se inizialmente nella finestrina c'era scritto UTF8, si scelga ISO8859-1, poi si clicchi di nuovo sulla finestrina e nella prima riga in alto si scelga di "ricaricare lo stesso file con la codifica appena scelta".
   (f) Il file viene ricaricato: se i rombi neri spariscono dalle righe precedentemente segnate, quella appena scelta è la codifica con cui era

stato scritto il file; se i rombi neri non spariscono ripetere scegliendo un altra codifica come indicato nel passo precedente. Si noti che le codifiche più frequenti sono ISO8859-1, MacRoman, Windows 1250, oltre a UTF-8 che è la codifica preimpostata di TeXworks e che sarebbe da preferire sempre.

(g) Se si vuole cambiare codifica (come si è supposto all'inizio di questa elencazione), si scelga la codifica desiderata nella solita finestrina e si salvi il file.

6. In questo modo si è scoperta la codifica del file iniziale e lo si è convertito nella codifica desiderata; se si dispone di TeXShop, si apra il file appena salvato e con i suoi pulsanti gli si faccia scrivere in testa al file la riga magica della codifica corrispondente a quella appena usata per la conversione. In questo modo non ci si dovrà più preoccupare di capire/indovinare la codifica ignota di quel file. Se non si dispone di TeXShop, si scrivano a mano le righe magiche seguendo la stessa sintassi mostrata nella pagina 670 e seguente, ricordando, se necessario, di specificare il nome della codifica usata al posto di `Isolatin`.

Quando ho cominciato a curare questo testo le opzioni per usare la codifica *utf8* erano appena agli inizi e non erano sufficientemente affidabili; perciò avevo sempre consigliato di usare la codifica *latin1*; pur non essendo completamente conforme con qualunque impostazione di sistema operativo e/o di *shell editor*, per scrivere in italiano e nella maggior parte delle lingue occidentali questa codifica va benissimo; oggi che la codifica *utf8* è a punto e stabile non ho dubbi nel raccomandare, più che consigliare, di usare questa codifica "universale".

Con la codifica di entrata *utf8* il file sorgente aumenta leggermente di dimensioni ma si è sicuri che, pur di usare uno *shell editor* recente che accetti la codifica di ingresso *utf8*, non esistono problemi di codifica a livello di tutti i programmi di composizione del sistema TEX. Va anche notato che le versioni aggiornate dei più moderni *shell editor* già dalla prima installazione sono preimpostati per lavorare con la codifica *utf8*.

A chi si avvicina al sistema TEX per la prima volta, oppure aggiorna la propria distribuzione del sistema TEX dopo anni di mancati aggiornamenti, consiglio di installare, se non lo sono già, uno o più degli editor TeXworks, TeXstudio, Texmaker, che sono multipiattaforma, e di impostarli, se non lo fossero già, per lavorare con la codifica di ingresso *utf8* e di mantenere sempre questa codifica. Se capitasse loro di dover aprire un file ricevuto da terzi, o scaricato dalla rete, che non fosse scritto con questa codifica, procedano come detto sopra per convertire il file alla codifica *utf8*.

Tuttavia la codifica *utf8* non garantisce di evitare problemi quando si esegue lavoro cooperativo, visto che i vari 'tastieristi' non necessariamente dispongono degli stessi *shell editor* operanti su macchine dello stesso genere, controllate dallo stesso sistema operativo. Bisogna sempre essere pronti a riconoscere i problemi della codifica di entrata e saperli risolvere. Questa raccomandazione è particolarmente indicata per coloro che si sono avvicinati al sistema TEX da

poco; oppure a coloro che aggiornano il loro strumenti software a versioni più moderne e non sono pronti ad affrontare questo genere di “novità”.

Capitolo 27

Come fa TEX a comporre le pagine

Sul Forum del GUIT compaiono spesso domande che rivelano la difficoltà degli utenti a padroneggiare il meccanismo di composizione asincrona del sistema TEX. Questo meccanismo, si badi bene, è sostanzialmente identico con qualunque dei motori di composizione del sistema, sia esso tex, pdftex, context, luatex, xetex, o gli altri motori meno conosciuti o meno usati. Per semplicità nel seguito ci riferiremo a pdftex, che insieme a lualatex, permette di sfruttare completamente la microgiustificazione, ma quanto verrà detto per questo motore di composizione vale sostanzialmente anche per gli altri, con o senza microgiustificazione.

In tutti i casi pdftex determina per ogni capoverso e per ogni pagina dei fattori di merito o di demerito tenendo conto di misure di 'bruttezza', nel gergo di TEX chiamata *badness*, delle penalità (*penalty*) e dei fattori di demerito (*demerit*) che vengono intercalati nel testo e nelle pagine da comporre, diciamo così, all'insaputa del compositore.

Il compositore inserisce fattori di merito o di demerito o penalità di vario genere nel file sorgente quanto usa comandi quali `\pagebreak`, `\clearpage` o `\finalhyphendemerits`, ma spesso questi comandi e i relativi parametri sono già presenti nei file di classe. In più pdftex determina altri fattori di bruttezza o di demerito per conto proprio. Il risultato di tutte queste operazioni è che il motore di composizione sceglie le divisioni in righe dei singoli capoversi e le divisioni in pagine dei singoli capitoli in modo da minimizzare la bruttezza complessiva dei capoversi oppure delle pagine.

L'algoritmo che determina questo minimo è decisamente complicato da un punto di vista del programma di pdftex, anche perché la ricerca dell'ottimo deve essere eseguita velocemente e con il massimo di efficienza e di efficacia.

27.1 Divisione dei capoversi

Cominciamo a riesaminare il procedimento di divisione dei capoversi in righe. Se ne è già parlato nel capitolo 25, ma ci torniamo sopra per esaminare meglio questo meccanismo.

27.1.1 La microgiustificazione

Cominciamo col ricordare che il capoverso inizialmente costituisce una sola lunga riga e il motore di composizione deve spezzare questa riga in monconi tutti della stessa lunghezza (ovviamente, tranne l'ultimo moncone che può essere più corto). Se è stato invocato il pacchetto *microtype* esiste una complicazione in più, perché la stessa lunghezza può essere raggiunta anche con i meccanismi di protrusione e di leggera deformazione orizzontale dei font che compongono ogni riga di testo, cosicché in prima battuta, prima, cioè, di applicare questi meccanismi raffinati, le righe di testo potrebbero essere considerate di uguale lunghezza entro una certa tolleranza, dell'ordine di grandezza di qualche centesimo della lunghezza da raggiungere; sarà poi compito delle macro del pacchetto *microtype* quello di allungare o restringere orizzontalmente il font presente nella riga aggiustandone la lunghezza per recuperare quei pochi punti percentuali che mancano o che eccedono rispetto alla lunghezza desiderata. In mancanza di questo pacchetto o della sua piena funzionalità, il compito di regolare la lunghezza della riga al valore desiderato spetta a pdftex che vi provvede allungando o accorciando solo la gomma interparola o altra eventuale gomma presente nella riga. Capito questo concetto, lasciamo da parte la microgiustificazione e concentriamoci sul meccanismo standard.

27.1.2 Il meccanismo generale

La lunga riga iniziale potrebbe già di per sé contenere delle penalità positive o negative: per esempio, ogni volta che si è usato un segno di legatura mediante la tilde, `~`, che inserisce nel testo uno spazio indivisibile, prima del quale non si può andare a capo, viene inserita una penalità 'infinita', che per TEX è un qualunque valore maggiore o uguale a 10 000. Oppure potrebbe essere stato inserito un comando `\\` che ordina di spezzare la riga in quel punto; e questo 'ordine' viene esplicitato inserendo nella lunga riga una penalità negativa infinita (un valore minore o uguale a −10 000). Inoltre la descrizione della lingua italiana richiede anche l'inserimento di un valore estremamente elevato di demerito se la penultima riga di qualsivoglia capoverso terminasse con una cesura, cioè se l'ultima parola della penultima riga di un capoverso fosse divisa in fin di riga.

pdftex va allora a cercare tutti i possibili punti di divisione in righe dell'intero capoverso, che per ora forma una sola lunga riga, associando ad ogni punto di divisione un valore di demerito. Questo valore di demerito viene determinato in base alle penalità associate al quel punto di divisione, ai demeriti conseguenti alla presenza di una cesura alla fine della riga, alla bruttezza della riga determinata

in base all'allungamento o accorciamento della gomma interparola e dell'altra gomma eventualmente inserita dal compositore, per esempio mediante dei *leaders*.

Per calcolare i demeriti associati ad ogni possibile punto di divisione pdftex deve determinare la bruttezza di ogni riga; questa è associata al fatto che la gomma complessiva presente nella riga sia stata allargata o accorciata molto o poco in relazione all'*allungabilità* o all'*accorciabilità* complessiva presente nella riga di cui si sta valutando la possibile fine. Sia z l'allungamento o l'accorciamento subito dalla gomma per raggiungere la lunghezza desiderata; z può essere positivo (allungamento) o negativo (accorciamento). Sia x l'allungabilità complessiva presente nella riga e sia y la sua accorciabilità. La bruttezza b è data da una espressione numerica di questo tipo:

$$b = 100 \times \begin{cases} \left(\dfrac{z}{x}\right)^3 & \text{se è avvenuto un allungamento} \\ \left(\dfrac{|z|}{y}\right)^3 & \text{se è avvento un accorciamento} \end{cases} \tag{27.1}$$

saturata al valore 10 000 se per caso il valore calcolato con l'espressione 27.1 risultasse maggiore.

Va notata l'asimmetria delle espressioni per l'allungamento e per l'accorciamento; è ovvio che l'accorciamento massimo di una riga non può superare l'accorciamento disponibile, quindi la bruttezza associata ad un accorciamento non supera mai il valore 100. Invece l'allungamento può essere maggiore dell'allungabilità, per cui la bruttezza associata ad un allungamento può essere molto grande. Si nota anche che se l'allungabilità o l'accorciabilità sono infinite, la bruttezza è nulla. La presenza della terza potenza fa sì che allungamenti o accorciamenti piccoli rispetto alle rispettive allungabilità o accorciabilità sono del tutto trascurabili, mentre diventano sempre più importanti quando questi rapporti sono vicini all'unità o la superano.

pdftex cerca di evitare che due righe adiacenti abbiano dei valori molto diversi di allungamento o di accorciamento, meno che mai una accorciata e la vicina allungata; considera le righe *decenti* se la loro bruttezza è inferiore al valore 12, mentre se lo eccede, misura la loro *compatibilità visiva* e se due righe adiacenti risultassero incompatibili aggiungerebbe dei demeriti a quel particolare punto di divisione.

In ogni caso calcolata la bruttezza di ogni riga e specificati i demeriti, pdftex deve determinare la bruttezza complessiva delle possibili divisioni in righe di un dato capoverso; per far questa somma calcola i demeriti associati ad ogni possibile punto di divisione e sceglie quella sequenza di punti di divisione che totalizza la minima somma di demeriti.

Intanto pdftex toglie dalla lista tutti i possibili punti di divisione ai quali siano associate penalità p maggiori o uguali a 10 000 e tutti i punti in cui la bruttezza sia superiore alla `\pretolerance` o `\tolerance` correnti; ricordiamo che questi due valori servono per valutare una prima possibile divisione in righe senza cesure, e poi una seconda divisione con le cesure in fin di riga; se la prima

non è andata a buon fine, cioè se tutti i possibili punti di divisione alla fine di ogni parola avevano bruttezza superiore al valore di `\pretolerance`, allora procede ad eseguire la seconda divisione con le cesure. In questo testo `\tolerance` vale 200 e `\pretolerance` vale 100.

Va notato ancora che a ogni riga che termina con una cesura viene attribuita una penalità indicata dal parametro `\hyphenpenalty`; se due righe consecutive sono entrambe terminate con una cesura, viene loro attributo un ulteriore demerito dato dal parametro `\doublehyphendemerits`. In questo testo questi demeriti sono dati rispettivamente dai valori 50 e 10000.

La gomma elastica fra le parole è una caratteristica del font in uso ma può fare uso del termine `\emergencystretch`, fissato in 3 em nel nucleo di LaTeX; questo valore viene usato solo all'interno dell'ambiente *sloppy*, che può venire usato in caso di emergenza quando un capoverso è tanto difficile da giustificare che, pur di non uscire dai margini, il compositore è disposto localmente a comporre un capoverso in modo *sloppy*, cioè mal fatto. L'ambiente *fussy* ristabilisce un valore di `\tolerance` piccolino (200), e rimette a zero il termine `\emegencystretch`. Fuori da questi due ambienti il valori di `\tolerance` e di `\emergencystretch` sono fissati nella classe in uso o fissati dall'utente. In questo testo essi normalmente valgono 200 e 0.0pt.

pdftex deve anche tenere conto della penalità `\linepenalty` legata al numero complessivo di righe in un capoverso; il valore predefinito per questa penalità è $l = 10$, ma il file di classe potrebbe definire un valore maggiore per cercare di mantenere i capoversi i più corti possibili. Il valore dei demeriti d associati ad ogni possibile punto di divisione, fra quelli rimasti nella lista, viene determinato con l'espressione seguente:

$$d = \begin{cases} (l-b)^2 + p^2 & \text{se } 0 \le p < 10\,000 \\ (l+b)^2 - p^2 & \text{se } -10{,}000 < p < 0 \\ (l+b)^2 & \text{se } p \le -10\,000 \end{cases} \tag{27.2}$$

L'espressione per i demeriti presenta una discontinuità che ha i suoi motivi, nei dettagli dei quali non scendiamo. Queste espressioni servono per dare una misura di demerito approssimativamente proporzionale alla somma del quadrato della bruttezza con il quadrato della penalità. Minimizzare la somma dei demeriti per un intero capoverso, vuol dire pressappoco minimizzare la bruttezza di ogni riga. Questo è appunto uno dei punti di forza di questo algoritmo che può venire ottenuto solo con la composizione differita eseguita dai vari motori di composizione del sistema TeX.

Esso è tanto preciso nel determinare il minimo demerito associato ad una particolare divisione in righe, che se un capoverso andasse ritoccato per aggiungere un virgola o una breve congiunzione, potrebbe succedere (e a chi scrive è successo di osservare questo fenomeno) che il capoverso così accresciuto nel numero delle lettere che contiene, può risultare una riga più corto rispetto a prima di eseguire la modifica: non è frequente, ma succede. Questo dipende dalla raffinatezza della ricerca del minimo eseguita dal motore di composizione.

Prima di elaborare ulteriormente il capoverso così creato, il processo di formazione dei capoversi inserisce dopo la prima riga del capoverso la penalità per le righe orfane e prima dell'ultima riga la penalità per le righe vedove. Queste penalità servono al processo di formazione delle pagine per evitare di eseguire i tagli di pagina lasciando righe orfane o vedove. Queste penalità solitamente non sono infinite; in questo testo `\widowpenalty` vale 3000, e `\clubpenalty` vale 3000.

Finito di scomporre il capoverso in righe, le scatole orizzontali che contengono ogni riga vengono accodate alla scatola 255 riservata all'uso di deposito temporaneo del testo composto[1]. La scatola 255 svolge quello che in inglese si chiama *galley proof*, una parte di testo da stampare come bozza, con il testo composto ma non impaginato. Ad ogni capoverso accodato alla scatola 255, viene anche attivato il meccanismo della formazione della pagina; se la scatola 255 contiene abbastanza materiale, la prima parte della scatola viene emessa nel file di uscita e quanto avanza serve per accodarvi il testo che via via viene ulteriormente composto e che verrà usato per creare le pagine successive.

27.2 Divisione della pagine

La divisione delle pagine avviene secondo gli stessi principi seguiti per la divisione dei capoversi, con alcune evidenti varianti. Mentre un capoverso da dividere in righe forma una lista orizzontale di caratteri, penalità, grumi di gomma, eccetera, un testo da dividere in pagine è costituito dalla lista verticale conservata nella scatola 255; questa lista è formata dalle righe già composte per il testo, dalle scatole che contengono, per esempio, singole espressioni matematiche, da penalità, da grumi di gomma verticale, ma non solo. Essa contiene anche dei segnaposti per inserire oggetti che sono stati conservati altrove: gli inserti delle note a piè di pagina, le note marginali, le figure, le tabelle e gli altri oggetti flottanti.

pdftex, nell'estrarre le righe da riversare in una pagina del file di uscita, deve tenere conto dei segnaposti e intercalare alle righe di testo quanto questi segnaposti segnalano. Per le note in calce deve tenere presente della riga dove appare il richiamo in modo da mettere la nota nella stessa pagina dove essa è richiamata; e se non ci riesce deve mettere una forte penalità per questo 'insuccesso'; deve mettere anche una penalità se la nota può cominciare nella pagina ma non ci sta tutta, per cui deve estrarne solo alcune righe per rimettere le righe che avanzano in una posizione utile per la pagina successiva, inserendovi anche una penalità per tenere conto di questo 'insuccesso'. Deve vedere se le note marginali stanno nella pagina o se devono essere spostate nella pagina successiva; deve controllare anche se le note marginali sono abbastanza separate

[1] In verità il capoverso così composto viene accodato alla *main vertical list* e la scatola 255 serve come buffer per le operazioni successive; la descrizione di questo paragrafo è volutamente semplificata mediante questa imprecisione, per non scendere in dettagli troppo tecnici; tuttavia il funzionamento vero della composizione delle pagine non è molto diverso da quanto descritto qui in modo semplificato.

le une dalle altre, ed eventualmente deve spostare i segnaposti per farcele stare, ma aggiungendo altre penalità. Deve esaminare le code degli oggetti flottanti, controllando che ci sia abbastanza posto per farne stare alcuni sulla pagina, e dove; se non possono stare sulla pagina deve lasciarli in coda, in modo da poterne differire il collocamento ad una pagina successiva; deve tenere conto dei comandi `\clearpage`, e simili, che hanno messo nella lista verticale speciali penalità negative minori di $-10\,000$, e che funzionano da codici per particolari azioni che non richiedono solo un salto di pagina, ma anche lo svuotamento forzato delle code degli oggetti flottanti. Determinata la divisione migliore, che generalmente non supera l'altezza della gabbia del testo, la procedura per la spedizione della pagina completa al file di uscita provvede a inserire la testatina e il piedino.

Come si vede, la formazione delle pagine è sotto molti aspetti più complessa della formazione dei capoversi, ma poi dopo aver deciso che cosa mettere nella pagina, il punto migliore in cui porre la divisione avviene con lo stesso algoritmo usato per i capoversi, nel senso che ora non ci sono due possibili scelte, quella senza cesure e quella con cesure in fin di riga, ma c'è una sola possibile scelta, quella che minimizza i demeriti complessivi della pagina.

Il calcolo della bruttezza avviene più meno nello stesso modo, solo che la gomma interparola è sostituita dalla gomma interlinea `\lineskip`, che però, solitamente, non ha nessuna componente di allungabilità o accorciabilità; della gomma prima della prima riga di ogni capoverso `\parskip`, che generalmente ha una lunghezza naturale nulla e una modesta allungabilità. Della gomma inserita dai comandi di sezionamento prima della loro prima riga, oltre che della penalità inserita dopo la loro fine, in modo da impedire un salto di pagina dopo il titolo (cosa che notoriamente avviene, invece, con i comuni word processor); della gomma presente prima o dopo gli oggetti fuori testo, dalle espressioni matematiche agli oggetti flottanti in genere. La formula della bruttezza è simile all'espressione 27.1 con le ovvie differenze di significato dei simboli presenti.

Invece i demeriti sono determinati diversamente; il loro valore rappresenta il prezzo che bisogna pagare se si interrompesse la pagina nel punto dove lo si calcola; essi sono perciò chiamati *costo* e vengono indicati con il simbolo c; il costo è tanto maggiore quanto peggiore risulta la pagina; per ogni possibile punto di divisione della pagina il costo associato vale:

$$c = \begin{cases} p & \text{se } b < \infty \text{ e } p \leq -10\,000 \text{ e } q < 10\,000 \\ b + p + q & \text{se } b < 10\,000 \text{ e } |p| < 10\,000 \text{ e } q < 10\,000 \\ 100\,000 & \text{se } b = 10\,000 \text{ e } |p| < 10\,000 \text{ e } q < 10\,000 \\ \infty & \text{se } (b = \infty \text{ oppure } q \geq 10\,000) \text{ e } p < 10\,000 \end{cases} \tag{27.3}$$

dove: b rappresenta la bruttezza, ed è un numero compreso fra 0 e 10 000, o 'infinito' quando la divisione corrisponde ad una pagina che supera l'altezza della gabbia del testo – questo non dovrebbe succedere mai, ma talvolta succede a causa di inserti troppo grandi –; p rappresenta la penalità; q rappresenta le `\insertpenalties`, cioè la somma delle penalità per le note differite o spezzate

fra pagine. La divisione finale della pagina avviene nel punto nel quale la somma dei costi associati ai singoli punti di divisione viene minimizzata.

27.3 Cosa fare se...

Come si vede, dunque, l'algoritmo di ottimizzazione della divisione in pagine della scatola 255, con i suoi annessi e connessi, è diverso da quello della divisione dei capoversi, anche se è ispirato da una filosofia simile. Si tratta sempre di scegliere il punto dove i costi sono minimi; se quello è il punto in cui essi sono minimi, significa che ogni altro punto di divisione è peggiore; quindi non c'è da stupirsi se qualche volta pdftex spezza le pagine diversamente da come noi ce lo immagineremmo, se cioè colloca gli oggetti flottanti diversamente da dove ce lo aspetteremmo.

Certo potremmo cambiare i criteri con i quali le parti della pagina possono venire utilizzate per gli oggetti flottanti e per il testo; certo potremmo cambiare le penalità associate alle righe orfane e vedove. E in effetti chi scrive ha spesso usato questi espedienti, ma con grande cautela. I valori preimpostati sono degli ottimi compromessi e cambiarli richiede giudizio e comprensione del meccanismo.

Vale la pena di raccogliere qui alcune informazioni esposte anche altrove.

27.3.1 Collocazione degli oggetti flottanti

pdftex mette gli oggetti flottanti dove lo consentono le condizioni imposte dai vari parametri specificati dalla classe e descritti approfonditamente nel paragrafo 29.10. Come ricordato, quei parametri e quelle frazioni di pagina possono venire cambiati un pochino, ma non troppo. Se si assegna, per esempio, una percentuale del 90% alla frazione di pagina destinata agli oggetti in testa alla pagina, si rischia di avere troppo poco spazio dedicato al testo, e se questo contiene titoli di paragrafi o elementi fuori testo come le espressioni matematiche, questa parte di testo non potrebbe stare nella restante frazione del 10%, e la pagina risulterebbe mozza oppure l'oggetto flottante potrebbe venire spostato ad una pagina successiva.

Se il compositore usa troppo spesso il parametro di collocamento `!` rischia di fare peggio di quanto LaTeX farebbe per conto suo. Si ricorda che questo specificatore non modifica l'osservanza dei parametri solo per la posizione dichiarata con il primo degli specificatori, ma li riguarda tutti. Usare la serie di specificatori `h`, `t`, `b`, `p` serve a poco, perché va a finire che se pdftex non può usare la posizione specificata con `h`, se è l'unica opzione specificata, allora l'oggetto mobile non può più essere collocato "lì" e rischia di restare in coda a lungo; eventualmente la coda degli oggetti flottati può venire svuotata d'autorità o alla fine di un capitolo o per effetto di un comando `\clearpage` (eventualmente passato come argomento a un comando `\afterpage`). È meglio evitare di specificare `h`; non è un caso che le posizioni predefinite in LaTeX sono proprio `t`, `b`, `p`, perché se esso non può andare in testa a 'questa' pagina, vi può andare in calce, oppure

può andare in testa o in calce in una pagina successiva, oppure quando vengano svuotate le code. Se l'oggetto è di grandi dimensioni, è meglio specificare solo `p`, cosicché alla prima pagina utile esso viene collocato in una pagina a sé stante (sempre che la coda non sia già intasata) e spedito al file di uscita.

Raramente capita che le code intasate restino tali via via che si aggiungono altri elementi in coda; quando si supera il numero di 56 oggetti in coda, i successivi vengono persi; sì, è vero, il programma emette un avviso, ma è una magra consolazione[2] Se si ha anche solo il dubbio che la coda possa essere intasata da un oggetto troppo grande per condividere una pagina con un po' di testo, è meglio verificare la cosa durante la lavorazione del documento emettendo un comando `\clearpage`, così da vedere che cosa ci sia in coda, e poi eventualmente cambiare i descrittori di posizione, emettere uno o più `\clearpage` passandoli al programma come argomenti di `\afterpage`; insomma in modo da superare le barriere messe dal programma contro una brutta composizione tipografica. Agendo con giudizio, si può aggiustare a mano quanto il programma non riesce a gestire da solo; ma questo va fatto in parte durante la lavorazione del documento e in parte alla fine per dare i tocchi finali al lavoro compiuto.

27.3.2 Le equazioni ingombranti

Le 'piccole' equazioni fuori testo, fra espressione matematica vera e propria e spazi sopra e sotto, impegnano l'equivalente di tre righe di testo; espressioni matematiche più complesse possono richiedere un numero di righe molto maggiore. Se una di queste espressioni, 'piccole' o 'grandi' che siano, compare in testa ad una pagina, è molto probabile che la pagina precedente sia stata dichiarata 'Underfull \vbox' da `pdftex`, perché ha dovuto stiracchiare troppo la gomma disponibile sulla pagina per non farla sembrare mozza; in realtà non è mozza, ma è brutta a causa dell'eccessivo allargamento dei contrografismi che separano le varie parti della pagina.

In questi casi non ci sono cure diverse dalle seguenti.

1. Modificare il testo della pagina stiracchiata, in modo che contenga *più* testo.
2. Modificare il testo della pagina stiracchiata, in modo che contenga *meno* testo.
3. Allungare la pagina stiracchiata usando il comando `\enlargethispage` con o senza asterisco, in modo che lo specchio di stampa sia un poco più grande così da contenere una o due righe in più.
4. Accorciare lo specchio di stampa della pagina stiracchiata, sempre usando `\enlargethispage` ma usando un allungamento negativo, così da richiedere uno stiracchiamento minore.
5. Usare il comando primitivo `\looseness` seguito da un numero intero positivo o negativo per allungare o accorciare, se possibile, capoversi scelti

[2] Con le modifiche recenti del nucleo di LaTeXsembra che non esista più un numero finito e relativamente piccolo del numero di oggetti flottanti presenti nelle code.

della pagina stiracchiata così da riempire meglio la pagina; 'capoversi scelti' vuol dire capoversi sufficientemente lunghi da avere molto spazio interparola su cui giocare per ottenere l'effetto desiderato. Il numero che si assegna al comando `\looseness` indica di quante righe allungare o accorciare un capoverso; a meno che non si tratti di un capoverso molto lungo, è difficile allungare o accorciare un capoverso di più di una riga. Tuttavia spesso uno o due capoversi allungati di una riga possono aggiustare una pagina in modo difficilmente riconoscibile a occhio nudo.

6. Inserire un comando `\newpage` prima di un comando di sezionamento che ha prodotto lo stiracchiamento della pagina precedente. Chi scrive trova comodo usare un comando condizionale per iniziare una nuova pagina: `\goodpagebreak` tronca la pagina se mancano alla fine fino a 4 righe di default, ma un diverso numero di righe che si può specificare come argomento al comando. La definizione, che ricorre ai comandi estesi dei programmi di composizione, è la seguente:

```
\newcommand\goodpagebreak[1][4]{%
\unless\ifdim\dimexpr \pagegoal-\pagetotal>#1\baselineskip
   \newpage\fi}
```

Il vantaggio di usare questo comando è che se si modifica il testo prima dell'eventuale salto di pagina, il comando può diventare inerte perché non serve più eseguire il salto pagina e non c'è bisogno di intervenire a mano.

È chiaro che questi provvedimenti, tranne quelli che prevedono l'aumento o la diminuzione del testo, sono provvedimenti da prendere solo nella fase finale di messa a punto dell'impaginazione, quando il testo è praticamente finito di comporre, altrimenti quegli aggiustamenti finirebbero per dover essere rivisti ripetutamente ad ogni modifica del testo che li precede.

Queste cose vanno viste con molta cura anche quando si eseguono degli inserti non flottanti, come una piccola figura contornata dal testo; questa è legata al particolare capoverso il cui testo le gira attorno e se la modifica del testo che lo precede lo sposta più in alto o più in basso, la figurina fissa rischia di andare a finire dove l'algoritmo di formazione delle pagine inserisce un salto pagina; la figurina potrebbe rimanere lì, penzoloni, alla fine della pagina, a fianco di un paio di righe del suo capoverso, che però proseguirebbe nella pagina successiva con la rientranza vuota inizialmente riservata per la figurina. Cose queste che sconsigliano fortemente di fare uso di piccole figure contornate dal testo, se non nel *primo* capoverso di un capitolo, perché non vi è nessun testo prima che possa spostare il primo capoverso più in basso.

27.3.3 I sezionamenti

Anche i titoli dei paragrafi e delle sezioni di rango successivo possono dare problemi. Si è detto e ripetuto che lo spazio verticale che segue un titolino è uno spazio 'indivisibile', uno spazio verticale *prima e dopo* del quale non

può avere luogo un salto di pagina; quindi il titolo del paragrafo è distanziato dalla prima riga del suo primo capoverso, ma non ne può venire staccato. Se avvenisse un salto di pagina dopo la prima riga del capoverso, essa rimarrebbe orfana; per evitare questo la classe definisce una penalità `\clubpenalty`, insieme alla sua compagna `\@clubpenalty`, che viene inserita dopo la prima riga di ogni capoverso, proprio per evitare le righe orfane; dunque, sebbene non sia strettamente impossibile (con le penalità predefinite) un salto di pagina dopo la prima riga, la seconda riga appare subito sotto alla prima riga; ma se il capoverso ha ancora solo una terza riga, se avvenisse un salto di pagina dopo la seconda riga, la terza riga resterebbe vedova; una forte penalità `\widowpenalty` viene inserita prima dell'ultima riga di ogni capoverso, proprio per evitare la formazione di righe vedove, e ci riesce quasi sempre con i valori predefiniti. Ma tutto ciò implica che la riga del titolino, seguita dal suo spazio e seguito ancora da almeno due righe di testo costituisce un insieme 'monolitico' di grandi dimensioni che o sta tutto in fondo alla pagina o viene spostato tutto all'inizio della pagina successiva (naturalmente se è a metà pagina il problema non esiste). I valori di `\clubpenalty` e di `\widowpenalty` sono generalmente fissati dai file di classe, o dai file di descrizione della lingua, ma il compositore può cambiarne i valori in modo accorto; i valori predefiniti di solito sono piuttosto severi, ma generalmente non sono valori infiniti: in questo testo `\clubpenalty` vale 10000, e `\widowpenalty` vale 3000.

Si noti che sono considerate candidate a diventare orfane e vedove anche la prima riga dopo una equazione fuori testo o l'ultima riga prima di una equazione fuori testo; quindi il problema delle righe orfane e vedove si pone anche con le equazioni fuori testo.

In questi casi si può agire come negli altri casi, allungando o accorciando il testo che precede il blocco 'monolitico'; volendo, trattandosi di un sezionamento si potrebbe inserire subito prima del comando di sezionamento un comando `\newpage`; oppure talvolta è preferibile usare il comando`\goodpagebreak` definito nel paragrafo precedente; è vero che se il comando agisce, lascia una pagina mozza, ma se si corregge il testo precedente questo comando potrebbe restare inefficace, grazie alla piccola penalità che esso inserisce nella lista verticale. Il comando di basso livello (primitivo) `\filbreak` è troppo forte e in caso di modifiche agirebbe anche lontano dal fondo della pagina. Invece il comando `\goodpagepreak` è perentorio, ma è legato allo spazio disponibile dal punto in cui vene eseguito alla fine dell'altezza della gabbia del testo; se il comando agisce, esegue il salto pagina a ragion veduta solo nel punto in cui esso è collocato.

27.4 Conclusioni

La casistica potrebbe allungarsi moltissimo. Però ogni qual volta ci si trovi con una pagina mal formata, con troppo spazio bianco, la conoscenza del meccanismo di divisione delle pagine e il gioco sottile dell'allungamento della gomma in competizione con le penalità, e con le dimensioni dei grandi oggetti, permette di

risalire alle cause e ad apportare le modifiche del caso per risistemare la pagina mal riuscita.

Capitolo 28

Trattamento degli errori

Quando il programma di composizione incontra nel file sorgente delle parti che non riesce ad interpretare o non riesce ad eseguire, non sa cosa fare e si ferma con un messaggio d'errore; spesso il messaggio di errore che accompagna la descrizione dove ha trovato la difficoltà è sufficientemente chiaro. Queste in generale sono situazioni semplici; altre volte gli errori sono molto subdoli e i messaggi sono troppo vaghi.

28.1 Errori ortografici nei nomi dei comandi

La situazione elementare, quella di aver scritto in modo errato un comando, è già stata descritta alla fine del capitolo 5, e non ci si torna sopra. Quando il lettore arriva a leggere questo capitolo estremamente tecnico, vuol dire che si è scontrato con un errore davvero grave, ma ha già acquisito abbastanza esperienza per affrontare le situazioni elementari.

28.2 Errore nella ricerca dei file

Nello stesso modo si risolvono errori del tipo "File not found"; accanto al messaggio appare il nome del file che il programma di composizione non avrebbe trovato. L'obiezione più frequente che ci viene in mente è: "Ma come?! Se il file è nella cartella di lavoro!". Prima di lasciarci andare all'irritazione del caso, conviene leggere con calma e attenzione il nome del file che il programma non ha trovato e ragionarci sopra.

1. Dal 2020 i programmi di composizione del sistema TEX sono in grado di leggere nomi di file che contengano caratteri diversi da quelli ASCII e spazi. Se accanto al messaggio d'errore il nome del file appare spezzato a cavallo di uno spazio, vuol dire che il compositore sta usando una versione datata

del sistema TEX. I casi sono due: o aggiorna la sua installazione, oppure toglie gli spazi dai nomi dei file.

2. Il file è effettivamente lì, ma il sistema operativo distingue le lettere maiuscole da quelle minuscole nei nomi dei file; tutti i sistemi UNIX hanno questa particolarità, solo i sistemi operativi di tipo Windows non fanno differenza fra `MyFile.tex` e `myfile.tex`. La soluzione è controllare bene che si sia chiesto al programma di leggere il file il cui nome sia scritto con la stessa ortografia di come è registrato sul disco.
3. L'estensione `.tex` *non deve mai essere specificata*; i programmi del sistema TEX appendono sempre l'estensione `.tex` ai nomi di file che ne siano privi, ma a quel punto è bene essere sicuri di non aver inserito spazi spuri, cioè è importante che scriviamo `\input{myfile}` e non `\input{␣myfile␣}`.
4. Anche i file grafici è bene che siano specificati senza estensione; non è obbligatorio, ma è conveniente. Però, per esempio, stabilito che i programmi di composizione non leggono i file con estensione `.gif`, non è possibile cambiare l'estensione `.gif` in `.png`; l'estensione non è una decorazione inutile di un file, ma dice che formato ha il suo contenuto; cambiare l'estensione non cambia il contenuto. Eppure questo è un tipico errore che viene commesso più spesso di quanto non si creda, e a cui il programma reagisce con un messaggio di "File not found".
5. Il file potrebbe essere protetto e accessibile solo al proprietario del file o all'amministratore della macchina; prima di poterlo aprire bisogna modificarne gli attributi per consentirne l'apertura.
6. Un file di classe, o un pacchetto, o un qualunque altro file di servizio potrebbe essere stato installato nella cartella giusta, ma è possibile aver dimenticato di aggiornare il database dei nomi dei file; il programma di composizione non può trovarlo in quanto non è elencato i quei database. Esiste un database dei nomi dei file per ogni albero in cui è suddiviso l'insieme dei file di lavoro del sistema TEX; generalmente i file delle distribuzioni ufficiali sono messi in alberi accessibili solo all'amministratore della macchina. Nelle macchine Windows questo concetto di utente privilegiato ad eseguire qualunque operazione esiste, ma è meno preciso di come è definito sulle macchine UNIX. È quindi possibile che l'utente pensi di fare cosa giusta ad aggiungere i propri file personali nelle stesse cartelle dove risiedono i file del sistema TEX. Questo invece sarebbe un grave errore, perché al primo aggiornamento dei file di sistema i file personali sparirebbero. I file personali vanno inseriti in una struttura ad albero conforme al TDS (*TEX Directory System*) ma la radice di questo albero deve essere in una zona personale. Nelle macchine Windows la cartella corrispondente a `Home` non era ben definita nelle versioni precedenti a Windows 7, ma sicuramente le versioni più recenti dei sistemi operativi creati da Microsoft da Windows 8 in poi hanno una sorta di Home personale in `Utenti\`⟨*NomeUtente*⟩`\`. Questa è la cartella dove radicare l'albero personale; solo con MiKTEX essa va elencata fra i suoi alberi, in modo che quando si apre la cartella `MiKTeX Settings` il tasto `Refresh databases` possa agire anche sull'albero personale.

Sulle macchine Linux, questa zona è la cartella `Home`, indicata con `~/`; sulle macchine Apple, la radice dell'albero personale deve essere in `~/Library/`. Se si tratta di una installazione TEX Live o MacTEX non è necessario aggiornare il database dei nomi dei file dell'albero personale, ma se si tratta dell'installazione MiKTEX, ad ogni modifica del contenuto dell'albero personale bisogna aggiornare il database dei nomi dei file come spiegato sopra.

Come si vede sono tutte soluzioni semplicissime che non spaventano nessuno che conosca bene la propria macchina e il proprio sistema operativo. È vero; qualche volta situazioni di questo genere mettono in imbarazzo, ma la soluzione è poi immediata.

28.3 Ciclo infinito

L'unico errore per il quale ogni programma del sistema TEX non emette messaggi è forse l'errore più grave, l'entrata in un ciclo infinito; ogni programma, classe, o pacchetto del sistema TEX è in grado di eseguire delle chiamate ricorsive, ciò che avviene quando una macro direttamente o indirettamente richiama se stessa; perché questa ricorsione abbia termine è necessario che lungo questa operazione ciclica qualcosa cambi e un test ne verifichi il cambiamento al fine di non ripercorrere il ciclo all'infinito. Se ciò non accade il programma continua a riciclare all'infinito; a parte il tempo perso a riciclare a vuoto, può succedere che il programma di composizione scriva qualcosa nel file di uscita o nei file ausiliari. Se ciò malauguratamente succedesse e l'operatore non fosse pronto a "uccidere" il programma, si finirebbe per saturare l'intero disco con danni non facilmente quantificabili, ma sicuramente danni seri al contenuto del disco e alle operazioni necessarie al suo ripristino.

Questo genere di cose possono succedere quando si scrivono delle macro ricorsive; generalmente queste macro servono per eseguire elaborazioni che non coinvolgono l'utente finale, ma interessano moltissimo a coloro che scrivono classi oppure pacchetti o anche soltanto alcune macro per il preambolo del proprio documento.

L'unica via per uscirne è essere molto tempestivi, prima che possano essere danneggiate le informazioni sul proprio hard disk; spesso l'editor dispone di un pulsante di emergenza; in TeXShop c'è un bottone Abort; in TeXworks il bottone verde per lanciare la compilazione, può essere cliccato e oltre a diventare rosso arresta il programma. Sugli editor specifici per le macchine Linux o ci sono analoghi bottoni, oppure basta eseguire le scorciatoie di tastiera come Ctrl + Y, o simili. Sulle macchine Windows la procedura è più elaborata e più lunga, in quanto bisogna arrestare il programma agendo sul Task Manager, che però solitamente non ha un bottone sempre disponile in una delle barre, e quindi si consuma del tempo prezioso per eseguire tutta la procedura di "uccisione"

del programma entrato in un ciclo infinito. Con le macchine con un sistema operativo di tipo UNIX (Linux e Mac) si può sempre agire dalla finestra comandi dando prima il comando `ps ax` per determinare il numero del processo e poi dare il comando `kill 8902` (se il numero del processo è stato rilevato essere 8902). Le macchine Mac hanno anche il menù "mela", in alto a sinistra nella barra superiore, con la voce "Force quit"; cliccando su questa voce si apre una piccola finestra di dialogo dove sono elencati i processi dell'utente che sono in esecuzione, e di lì è facile individuare quella componente di TEX che sta facendo capricci; la si può selezionare e "uccidere".

28.4 Gruppi aperti

Un altro errore molto infido è la mancanza della parentesi chiusa di un gruppo; talvolta questa parentesi mancante provoca solo un avviso scritto verso la fine del file `.log` che spiega che il programma è terminato mentre si era dentro un gruppo; il messaggio spiega anche a quale livello di annidamento si trovava quel gruppo; qualche volta riesce a dire anche in quale riga di quale file quel gruppo era stato aperto. Talvolta il gruppo era stato aperto dentro un file incluso nel flusso di elaborazione mediante il comando `\include`; in questo caso si accorge che il gruppo non è chiuso quando ha finito di leggere il file incluso, ma il messaggio non riesce a dire altro che c'è un gruppo non chiuso e indica solo la riga del master file dove c'è il comando `\include`. È una informazione minima, ma almeno si riesce a sapere quale file il programma stava leggendo quando è arrivato alla fine con il gruppo ancora aperto.

Questo tipo di errore è relativamente facile da scovare; la maggior parte degli *shell editor* sono in grado di riconoscere le graffe appaiate; alcuni, in base a queste informazioni riescono a disegnare a lato della schermata dell'editor una struttura di linee che descrive graficamente la struttura del file e dei gruppi che contiene, siano essi ambienti delimitati dai comandi `\begin` e `\end`, siano essi gruppi delimitati dalle graffe. Chi scrive trova più comodo il funzionamento di TeXShop, che non ha la barra laterale; in questo *shell editor*, e in diversi altri per altri sistemi operativi, cliccando su una graffa l'editor cerca la sua compagna ed evidenzia tutto il gruppo con un colore di sfondo diverso; sia che il gruppo sia breve, sia che coinvolga diverse schermate del file sorgente, il brusco cambiamento di colore indica subito l'esistenza di un gruppo; al massimo potrebbe essere stato chiuso nel punto sbagliato, ma almeno il gruppo esiste. Quando invece il gruppo non è chiuso, non succede nient'altro e l'errore è trovato, basta solo andare a cercare il punto giusto per porre la parentesi chiusa mancante.

L'errore infido della mancata chiusura di un gruppo può talvolta essere molto difficile da scovare in un lungo documento non suddiviso in file da includere; anche in un documento suddiviso in diversi file inclusi può essere particolarmente dannoso se la mancanza del gruppo provoca messaggi d'errore incomprensibili e che non si riesce a mettere in relazione a quello che si crede di avere scritto dentro il file sorgente. In questo caso si può procedere col metodo della dicotomia;

si sposta la schermata dell'editor a metà del file (unico o incluso), non dentro un ambiente, ma prima o dopo l'apertura o, rispettivamente, la chiusura di un ambiente, e lì si inserisce una riga contenente solo

```
\end{document}
```

A questo punto si lancia il programma di composizione; se l'errore continua a presentarsi, la mancanza della chiusura del gruppo è nella prima metà del file; altrimenti è nella seconda metà. A questo punto si toglie la precedente specificazione di fine documento, lasciando al suo posto un contrassegno di posizione: potrebbe essere quella riga stessa dopo averla commentata. Poi la si inserisce nuovamente a metà della parte incriminata e si ripete il processo quante volte occorre fino a restare con una parte di file di poche decine di righe dove dovrebbe essere più facile trovare che cosa manca.

Si tratta di soluzioni estreme? Certo, quando non c'è nessuna informazione utile per capire la natura dell'errore bisogna ricorrere a soluzioni estreme; ammetto di averle dovute usare molto raramente ma in quasi quarant'anni d'uso di LaTeX potranno essere state una mezza dozzina di volte. Certo mi capitava più facilmente quando la distribuzione del sistema stava su un paio di floppy da $5^1/_4$ J e la documentazione era virtualmente inesistente; ora sono anni che non mi capita più, grazie sia alla migliore diagnostica e all'esperienza accumulata.

28.5 Mancata apertura di un gruppo

Manca sempre una graffa, questa volta aperta, mentre è presente la graffa chiusa. Il metodo di cliccare su graffe aperte o chiuse e controllare quale parte del file sorgente viene evidenziata funziona sempre, ma prima bisogna capire che si tratta di una graffa aperta mancante. Spesso il messaggio d'errore fa riferimento ad una graffa chiusa ed informa l'utente che:

```
./errori.tex:55: Too many }'s.
l.55 ... graffa chiusa}
                        ed informa l'utente che:
?
```

Si capisce subito di che cosa si tratta e che l'errore non è grave, per cui si può tranquillamente lasciar continuare la compilazione fino in fondo; bisogna però ricordarsi che la riga 55 del file `errori.tex` (come dice il messaggio) contiene una graffa chiusa priva di una precedente compagna graffa aperta e l'erroretto va comunque corretto.

28.6 Definizioni errate

Talvolta la graffa aperta mancante cade in una posizione insolita dove la corrispondente graffa chiusa ha un senso per il programma interprete. La graffa

chiusa non viene rilevata come erronea, e l'interpretazione continua fino ad un punto dove il programma non capisce più niente ed emette un messaggio d'errore che non ha nulla a che fare con il punto dove manca effettivamente la graffa aperta. Si consideri il codice seguente:

```
...
\newcommand\mycommand{%
\ifnum ....
    \def\RCi{\relax}\let\RCuno\RCi
    \def\RCii\relax}\let\RCdue\RCii
    \def\RCiii{\relax}\let\RCtre\RCiii
    ...
\else
    ...
\fi
...
}
```

Viene usato un test con la sintassi primitiva di TEX; la mancanza della graffa aperta nella definizione di `\RCCii` fa sì che il test iniziato con `\ifnum` non si chiuda all'interno della definizione. Se, usando TeXShop, avessimo cliccato sulla parentesi che apre la definizione di `\mycommand` avremmo trovato che questo "gruppo" si chiude dopo il comando `\relax` che segue la definizione di `\RCii` e ci saremmo accorti subito dell'errore. Non si è fatto questo controllo e si è mandato in compilazione il documento contenente queste definizioni. Poiché le macro contenute dentro le definizioni "normali" non vengono sviluppate (eseguite), il programma di composizione ci avrebbe avvisato della presenza di un "extra `\else`"; se si fosse forzato il programma a proseguire, si sarebbe avuto un altro avviso di errore per la presenza di un "extra `\fi`"; si sarebbero avute due segnalazioni di errore causati dalla mancanza di una graffa aperta, ma i due messaggi d'errore avrebbero portato l'attenzione sui comandi condizionali, non sulle graffe; se si fosse forzato ancora il programma a proseguire, finalmente esso avrebbe avvisato della presenza di "Too many }'s" e finalmente si sarebbe capito che mancava una graffa aperta o che c'era una graffa chiusa di troppo forzando a rivedere le graffe e a trovare finalmente la mancanza della graffa aperta.

28.7 File personali che provocano conflitti

In un file di classe avevo scritto il seguente codice:

```
\AfterEndPreamble{%
   \@ifundefined{virgola}{%
   \DeclareMathSymbol{\virgola}{\mathpunct}{letters}{"3B}
   \DeclareMathSymbol{\virgoladecimale}{\mathord}{letters}{"3B}}{}
}
```

Volevo evitare di ridefinire alcune assegnazioni eventualmente eseguite in altri file letti nel preambolo. Il lettore può riconoscere parte del codice illustrato nel capitolo 19 a proposito della virgola intelligente. Quel tipo di codice appare già nella mia versione personale del file di descrizione della lingua italiana `italian.ldf` letto da *babel* quando specifico la lingua italiana. Il mio errore consiste proprio nell'aver modificato a titolo personale un comando avente lo stesso nome di un comando presente in un file del sistema TEX che non deve mai essere modificato, mai e poi mai, per nessun motivo. "Mal me ne incolse."

Infatti la documentazione del pacchetto *etoolbox*, col quale è definito il comando `\AfterEndPreamble`, specifica chiaramente che i comandi contenuti nell'argomento di questo comando vengono sviluppati dopo che che è cessato il preambolo (come d'altronde il nome dice con grande chiarezza) e che quindi questi comandi non possono contenere altri comandi usabili solo nel preambolo. Il comando `\DeclareMathSymbol` è appunto uno di questi comandi a validità limitata che possono essere usati solo nel preambolo.

Anche se avevo recepito questo avvertimento, non l'avevo evidentemente interiorizzato e avevo scritto il codice suddetto che funzionava perfettamente e non ricevevo nessun messaggio d'errore. Tranquillo della bontà del codice, avevo mandato il file ad un amico; al quale la cosa non funzionava e riceveva il messaggio d'errore che:

```
... \DeclareMathSymbol
                      may be used only in the preamble
... See the LaTeX Companion for explanation
...
?
```

L'amico mi scriveva dicendomi "ricevo questo messaggio d'errore, ti mando il file `.log`...", e io gli rispondevo "Non riesco a replicare il tuo errore; dammi più dettagli...". Insomma fra noi c'era un dialogo fra sordi, ma il sordo ero io che avevo completamente rimosso l'avvertimento del pacchetto *etoolbox* e non lo collegavo con il messaggio piuttosto esplicito dell'amico.

Il problema ere causato dal mio file personalizzato `italian.ldf` che definiva il comando `\virgola` per cui il codice incriminato veniva saltato grazie al test `\@ifundefined`; ho fatto perdere al mio amico una buona settimana di botta e risposta inconcludenti per colpa mia; finché finalmente mi si è accesa la luce; ho capito il problema e ho potuto rimediare con mille scuse. Per l'ennesima volta ho commesso il "peccato" di ridefinire un comando di sistema e per l'ennesima volta ne ho pagate le conseguenze. Ecco perché, tra l'altro, insisto sempre sulla necessità di non commettere un simile errore. Qui almeno aggiungo la raccomandazione di non essere ottusi, come lo sono stato io, a non cogliere il significato vero dei messaggi d'errore.

28.8 Tracciare l'operato del programma

Il sistema TEX non è così rozzo come potrebbe apparire da quanto descritto nei paragrafi precedenti di questo capitolo; quei paragrafi descrivono situazioni estreme alle quali un linguaggio interpretato, come il linguaggio TEX, talvolta non può far fronte.

TEX invece è dotato di un certo numero di comandi primitivi, alcuni dei quali richiedono degli argomenti e tra i pacchetti LATEX ce ne è uno che svolge questo compito molto bene, anche se i puristi preferiscono gestire individualmente i singoli parametri di tracciamento.

28.8.1 I comandi primitivi di tracciamento

I parametri di tracciamento sono descritti qui di seguito; va notato che sono quantità a cui bisogna assegnare un valore positivo; quando si assegna il valore zero diventano inefficaci: la sintassi per tutti è del tipo seguente:

\⟨*comando di tracciamento*⟩ = ⟨*numero*⟩

`\tracingonline` serve per emettere le informazioni di tracciamento sulla console; lo si può usare anche quando si lancia il programma di composizione attraverso uno *shell editor*, ma sarebbe meglio evitarlo specialmente per tracciati lunghi; la memoria della console è relativamente limitata e si rischia di perdere alcune parti del tracciato.

`\tracingmacros` con un valore positivo, meglio se pari a 2, mostra lo sviluppo delle macro via via che queste vengono sviluppate.

`\tracingstats` con un valore positivo mostra l'utilizzazione delle memorie del sistema TEX.

`\tracingparagraphs` serve per mostrare i calcoli eseguiti quando il programma compone un capoverso spezzandolo in righe possibilmente di uguale lunghezza.

`\tracingpages` serve per mostrare i calcoli eseguiti quando il programma spezza la scatola 255 per estrarre le pagine da accodare al file di uscita inserendovi dentro le note e gli oggetti mobili.

`\tracingoutput` serve per mostrare le operazioni che il programma esegue quando deve spedire una pagina composta al file di uscita.

`\tracinglostchars` serve per mostrare quello che succede quando si invoca l'uso di un particolare carattere (glifo) ma il font non lo contiene.

`\tracingcommands` serve per mostrare quello che il programma di composizione fa quando deve eseguire un comando.

`\tracingassigns` mostra quando viene assegnato un valore ad un registro di conteggio di numeri interi, di lunghezze, di lunghezze elastiche, eccetera.

`\tracingrestores` serve per mostrare che cosa viene ripristinato quando il programma di composizione esce da un gruppo.

`\showboxbreadth` indica il massimo numero di oggetti da mostrare in ciascun livello di inscatolamento durante il tracciato.

`\showboxdepth` mostra il massimo livello di inscatolamento da mostrare nel tracciato.

Chi scrive spesso si è servito della coppia di specificazioni:

```
\tracingcommands=2\tracingmacros=2\relax
⟨righe del file sorgente da tracciare⟩
\tracingcommands=0\tracingmacros=0\relax
```

Il difetto è che l'informazione è molto, forse troppo, abbondante; in particolare quando vengono selezionati dei font, quando si compone della matematica, quando si traccia la chiusura di un documento, quando le righe da tracciare coinvolgono un fine pagina e la spedizione di una pagina all'uscita.

Qualunque esempio, anche minimo che si potrebbe fare per l'uso di questi comandi impiegherebbe diverse pagine e non si ritiene che sia di nessuna utilità senza un commento riga per riga, che risulta in realtà superfluo visto che ogni riga del tracciamento è autoesplicativa. Se esiste qualche difficoltà, questa sta nell'interpretazione delle informazioni e nell'abilità del lettore di saltare le parti che non interessano. Tuttavia si impara; la lettura del tracciamento non è spiegata da nessuna parte, ma tutti quelli che hanno eseguito e letto dei tracciati li hanno compresi, certo con pazienza e con molta determinazione, ma ci sono riusciti.

28.8.2 Il pacchetto *trace*

Il pacchetto *trace* sembra poco flessibile, ma sostanzialmente traccia tutto quello che può essere tracciato tranne la selezione dei font, la costruzione delle parti matematiche, la costruzione dei capoversi e la costruzione delle pagine, delle quali espone solo la riga finale. Se si usano male i suoi comandi potrebbe succedere che venga tracciato anche l'operato della routine di uscita, quella che emette nel file di uscita una pagina definitivamente composta con il testo, le note al piede e le note marginali, le eventuali figure e quant'altro venga eseguito durante l'accodamento di una pagina completata al file di uscita.

I comandi per attivare o disattivare il tracciamento sono molto prevedibilmente `\traceon` e `\traceoff`; essi vanno messi a cavallo della parte che si vuole tracciare e che produce messaggi d'errore; per coloro che creano classi, pacchetti o macro personali, possono servire per scoprire perché le macro accuratamente programmate e realizzate non fanno quello che si vorrebbe che facessero. Se si dovesse mettere `\traceoff` proprio appena prima della chiusura di un gruppo, lo si può omettere, perché il comando `\traceon` inserito dentro lo stesso gruppo cessa il suo effetto alla chiusura del gruppo stesso.

In effetti `\traceon` serve per impostare i parametri di tracciamento in questo modo:

```
\@tracingtrue
```

```
\tracingstats=2
\tracingpages=1
\tracinglostchars=1
\tracingparagraphs=1
\errorcontextlines=\maxdimen
\tracingoutput=1
\showboxbreadth=\maxdimen
\showboxdepth=\maxdimen
\errorstopmode
\tracingmacros=2
\tracinggroups=1
\tracingrestores=1
\tracingcommands=2
\tracingassigns=2
\tracingonline\tracingonline@p
```

Il comando opposto `\traceoff` rimette a zero tutti quei parametri. Si noti che `\maxdimen` rappresenta la massima lunghezza trattabile dai programmi del sistema TEX; visto in termini di "scaled points", se si assegna questo valore ad un contatore intero, indica il massimo numero intero (diviso per quattro) gestibile dai programmi del sistema TEX[1]; di fatto assegnare questo valore a `\showboxbreadth` e a `\showboxdepth` equivale ad assegnare un valore talmente grande da poterlo considerare "infinito". `\@tracingtrue` impone il valore `true` alla variabile booleana `@tracing`. `\tracinngonline@p` è una macro che può contenere il valore 0 o 1 a seconda delle opzioni con cui viene chiamato il pacchetto *trace*.

Questi due comandi, oltre ad "accendere" il tracciamento e a "spegnerlo", fanno sì che le informazioni del tracciamento vengano scritte nel file `.log`. Terminata l'esecuzione del programma di composizione, si può aprire anche con lo stesso *shell editor* con cui si compone il file sorgente dei documenti, quel file `.log` e leggere tutto quello che ha fatto il programma nella parte tracciata.

Di ogni comando che viene eseguito viene mostrata la definizione e quali argomenti vengono passati al comando; poi uno per uno vengono elencati i comandi primitivi oppure vengono "espanse" le macro e viene mostrato su quali argomenti lavorano; si procede così fino alla fine della parte da tracciare, senza mostrare se non qualche breve parte relativa ai cambiamenti di font, quel tanto che basta per sapere quale comando sia stato eseguito e quale font in effetti sia stato selezionato; vengono mostrate tutte le aperture di gruppi con i loro livelli e le corrispondenti chiusure; come vengono ripristinate le macro o i contatori che abbiano ricevuto altre definizioni o altri valori all'interno di ogni gruppo; quali valori avessero le macro oggetto di test di vario genere; quali siano gli esiti dei test e quale biforcazione si sia presa alla fine del test; che cosa indicassero i token impliciti; che cosa contenessero le macro oggetto dei comandi `\expandafter`.

[1] Il registro `\maxdimen` vale 16383.99998pt; assegnando `\maxdimen` a un contatore intero, questo vale 1073741823, più di un miliardo.

Insomma per filo e per segno ogni possibile azione eseguita dal programma di composizione, salvo i dettagli meno interessanti, come per esempio la selezione dei font e poche altre cose; se interessassero queste informazioni si possono usare i comandi di tracciamento descritti nel paragrafo precedente.

La descrizione di queste cose è molto minuziosa; il file `.log` rischia di essere molto, molto grande; la sua lettura molto faticosa; la sua comprensione molto difficile finché non si è presa la mano. È una operazione da cuori molto forti. Ma anche con i comandi primitivi si hanno situazioni dello stesso genere, talvolta ancora più dettagliate. Da quando parti del nucleo di LaTeX sono programmate con il linguaggio LaTeX 3, il tracciamento è diventato ancora più lungo; tuttavia imparando a saltare le parti di minor interesse, le informazioni che si ottengono sono di solito risolutive: dipende da chi legge e da come legge.

Mettere qui un esempio vorrebbe dire scrivere un altro libro. Vi rinuncio. Tuttavia non esistono altri metodi per imparare a comprendere il tracciato contenuto nel file `.log` se non quello di provarci e di riprovarci; dopo un paio di volte si fa luce nella nostra visione dell'operato del programma di composizione e la lettura del tracciato non è più così difficile come le prime volte: lungo sì, noioso sì, ma oscuro non più.

Vanno usate alcune piccole astuzie: bisogna stare attenti a non inserire le due macro di accensione e spegnimento dentro una macro che viene usata centinaia di volte ma mettere i due comandi suddetti a cavallo di un breve brano di file sorgente che contenga una sola volta l'esecuzione di quella macro. Bisogna evitare tratti di testo sorgente che contengano numerosi cambiamenti di font; anche se per questi viene presentato solo un riassunto, se i cambiamenti sono numerosi, i riassunti sono numerosi e alla fine si perde di vista quello che interessa. Bisogna ovviamente, trovato l'errore o la causa di un malfunzionamento di una macro, commentare o cancellare quei due comandi, altrimenti il tracciamento continua ad essere eseguito rendendo difficoltosa la lettura di un eventuale altro tracciamento in un altro punto del file sorgente. Insomma si impara ad usare correttamente la funzionalità di tracciamento e a leggerne soltanto le parti che interessano. Diventa un aiuto prezioso, specialmente se si scrivono macro.

Infine `\errorstopmode` imposta il modo di funzionamento del programma in modo che si arresti ad ogni errore; spesso i programmi di composizione vengono lanciati con altre opzioni, per esempio `nonstopmode` che possono risultare più veloci nell'esecuzione; chi scrive preferisce sempre lavorare in modalità `errorstopmode`, cosi può rendersi conto subito degli errori commessi; avendo disponibile la schermata dello *shell editor* può nella maggior parte dei casi spostarsi alla riga segnalata come erronea, correggere subito quella riga nel file sorgente; contemporaneamente, alla richiesta del programma di composizione, egli introduce l'informazione corretta servendosi della lettera di codice `i`, molto comoda, appunto, per correggere subito certi errori di ortografia nei comandi, ma tale da poter proseguire subito, generalmente fino alla fine della compilazione dell'intero file.

28.9 Costruzione dei capoversi e delle pagine

Il tracciamento della costruzione dei capoversi e delle pagine è richiesto quando non si capisce bene perché un capoverso venga composto male o diversamente da come ce lo aspetteremmo; lo stesso vale per la costruzione delle pagine da spedire in uscita.

A differenza degli altri comandi di tracciamento descritti nel paragrafo precedente, questi comandi sono descritti approfonditamente nel TEXbook. I comandi in questione sono `\tracingparagraphs=1` e `\tracingparagraphs=0` per accendere e spegnere il tracciamento della costruzione dei capoversi al livello di dettaglio 1; analogamente per il tracciamento della costruzione delle pagine `\tracingpages=1` e `\tracingpages=0`.

In questo capitolo si sono dati i comandi per il tracciamento delle pagine durante il corpo di pagine precedenti (in formato B5). I messaggi ottenuti nel file `.log` sono ridotti all'osso:

```
[610]
% t=470.0 plus 9.0 minus 8.0 g=466.0 b=12 p=0 c=12#
% t=482.0 plus 10.0 minus 8.0 g=466.0 b=* p=3000 c=*
[611]
%% goal height=466.0, max depth=5.0
% t=10.0 g=466.0 b=10000 p=3000 c=100000#
% t=22.0 g=466.0 b=10000 p=0 c=100000#
% t=34.0 g=466.0 b=10000 p=0 c=100000#
% t=46.0 g=466.0 b=10000 p=0 c=100000#
% t=58.0 g=466.0 b=10000 p=0 c=100000#
% t=70.0 g=466.0 b=10000 p=0 c=100000#
% t=82.0 g=466.0 b=10000 p=0 c=100000#
% t=94.0 g=466.0 b=10000 p=0 c=100000#
% t=106.0 g=466.0 b=10000 p=0 c=100000#
% t=118.0 g=466.0 b=10000 p=0 c=100000#
% t=130.0 g=466.0 b=10000 p=3000 c=100000#
% t=142.0 g=466.0 b=10000 p=0 c=100000#
% t=154.0 plus 1.0 g=466.0 b=10000 p=3000 c=100000#
% t=166.0 plus 1.0 g=466.0 b=10000 p=0 c=100000#
% t=178.0 plus 1.0 g=466.0 b=10000 p=0 c=100000#
% t=190.0 plus 1.0 g=466.0 b=10000 p=3000 c=100000#
% t=202.0 plus 1.0 g=466.0 b=10000 p=0 c=100000#
% t=214.0 plus 2.0 g=466.0 b=10000 p=3000 c=100000#
% t=226.0 plus 2.0 g=466.0 b=10000 p=0 c=100000#
% t=238.0 plus 2.0 g=466.0 b=10000 p=0 c=100000#
% t=250.0 plus 2.0 g=466.0 b=10000 p=0 c=100000#
% t=262.0 plus 2.0 g=466.0 b=10000 p=0 c=100000#
% t=274.0 plus 2.0 g=466.0 b=10000 p=0 c=100000#
% t=286.0 plus 2.0 g=466.0 b=10000 p=0 c=100000#
```

```
% t=298.0 plus 2.0 g=466.0 b=10000 p=0 c=100000#
% t=310.0 plus 2.0 g=466.0 b=10000 p=0 c=100000#
% t=322.0 plus 2.0 g=466.0 b=10000 p=0 c=100000#
% t=334.0 plus 2.0 g=466.0 b=10000 p=0 c=100000#
% t=346.0 plus 2.0 g=466.0 b=10000 p=0 c=100000#
% t=358.0 plus 2.0 g=466.0 b=10000 p=0 c=100000#
% t=370.0 plus 2.0 g=466.0 b=10000 p=0 c=100000#
% t=382.0 plus 2.0 g=466.0 b=10000 p=3000 c=100000#
% t=394.0 plus 2.0 g=466.0 b=10000 p=0 c=100000#
% t=406.0 plus 3.0 g=466.0 b=10000 p=3000 c=100000#
% t=418.0 plus 3.0 g=466.0 b=10000 p=0 c=100000#
% t=430.0 plus 3.0 g=466.0 b=10000 p=0 c=100000#
% t=442.0 plus 3.0 g=466.0 b=10000 p=3000 c=100000#
% t=454.0 plus 3.0 g=466.0 b=6396 p=0 c=6396#
% t=466.0 plus 4.0 g=466.0 b=0 p=3000 c=3000#
% t=478.0 plus 4.0 g=466.0 b=* p=0 c=*
[612]
```

Possiamo leggere i risultati così:

1. Per la pagina indicata qui col numero `[611]` si legge che il totale di altezza al momento di spedire la pagina all'uscita è di 470,0 pt con la componente di allungamento totale nella pagina di 9,0 pt e una componente di accorciamento di 8,0 pt; l'altezza della pagina da raggiungere (il *goal*) è di 466,0 pt; quindi bisogna accorciare l'altezza della pagina di soli 4,0 pt, abbondantemente rientrante negli 8,0 pt di accorciamento disponibile. La bruttezza è perciò molto piccola (vale solamente 12) non ci sono penalità di nessun genere, il costo per l'emissione di quella pagina è quindi pari alla bruttezza e vale 12; un costo minimo che indica che la fine della pagina 611 è avvenuto nel migliore dei modi. Se si andasse ad una riga dopo, la bruttezza diventerebbe "infinita" (`b=*`) e il costo infinito.
2. Per la pagina `[612]` si legge il costo riga per riga: alla penultima riga della pagina si sono raggiunti 454,0 pt, con pochissimo allungamento accumulato, di soli 3,0 pt e non si dispone di nessuna componente di accorciamento. L'altezza della pagina da raggiungere è di 466,0 pt (come prima, evidentemente) ma i 12,0 pt mancanti devono essere recuperati dal piccolissimo allungamento disponibile di soli 3,0 pt; dunque la componente di bruttezza è piuttosto pesante, 6396, ma non infinita; per cui, in mancanza di penalità, il costo per emettere la pagina è piuttosto alto: 6396. In effetti quello sarebbe il costo per emettere la pagina una riga prima della riga orfana in fondo alla pagina. Quella riga in effetti è alta esattamente 12,0 pt, quindi il punto scelto dal programma per emettere la pagina non ha nessuna bruttezza, ma ha una penalità positiva pari alla penalità di default per le righe orfane pari a 3000, quindi abbastanza minore di quella corrispondente alla bruttezza raggiunta alla penultima riga. La pagina, quindi è stata

spezzata nel punto di minimo costo (c=3000) mentre una riga dopo si sarebbe ottenuto un costo grandissimo.

Si noti che avendo caricato il pacchetto `trace` *in questo testo il tracciamento delle pagine cessa con una sola informazione raggiunta verso la fine della pagina nel punto in cui si è specificato il comando di tracciamento e cessa da solo senza bisogno di specificare un nuovo comando con il valore nullo.*

Invece per tracciare il processo di composizione della formazione del capoverso precedente si ha il tracciato seguente, decisamente più complicato da decifrare:

```
@firstpass
[]\T1/lmr/m/it/10
Si noti che avendo caricato il pacchetto \T1/lmvtt/m/sl/10
trace[] \T1/lmr/m/it/10 in questo testo il tracciamento
@ via @@0 b=0 p=0 d=100
@@1: line 1.2 t=100 -> @@0
delle pagine cessa con una sola informazione raggiunta
 verso la fine della pagina
@ via @@1 b=0 p=0 d=100
@@2: line 2.2 t=200 -> @@1
nel punto in cui si è specificato il comando di tracciamento
e cessa da solo
@ via @@2 b=23 p=0 d=1089
@@3: line 3.1 t=1289 -> @@2
senza
@ via @@2 b=0 p=0 d=100
@@4: line 3.2 t=300 -> @@2
bisogno di specificare un nuovo comando con il valore nullo.
@\par via @@3 b=0 p=-10000 d=100
@\par via @@4 b=0 p=-10000 d=100
@@5: line 4.2- t=400 -> @@4
```

Il tracciato comincia con `@firstpass` che indica che le informazioni mostrate indicano la ricerca dei punti migliori di divisione in righe al primo passaggio, senza cioè usare punti di cesura, ma cercando i fine riga solo fra gli spazi interparola. Vi sono indicati i cambiamenti di font nelle righe via via prese in considerazione.

Il primo fine riga con un demerito totale pari a 100 viene trovato dopo la parola “tracciamento”; analogamente il secondo fine riga viene trovato dopo la parola “pagina” di nuovo con un demerito pari a 100; nella terza riga viene provata l’interruzione dopo la parola “solo”, ma i demeriti ammontano a 1089; invece andando avanti e prendendo in considerazione anche la parola “senza” i demeriti ammontano solo al valore 100; questo è quindi accettato come terzo fine riga possibile. Il quarto fine riga avviene alla fine del capoverso dove la penalità vale -10 000, e quindi non ci sono problemi nel terminare la riga alla fine del capoverso (ovviamente!). Nelle ultime due righe vengono riassunti i vari demeriti totali raggiunti attraverso i vari possibili punti di fine riga, e viene

scelto il percorso (segnato dai vari segnaposto `@@1, @@2, ecc.`) che porta al minimo della somma totale dei demeriti pari a 400. Ovviamente in capoversi più complessi, dove sia necessaria anche la seconda passata per cercare i possibili punti di cesura usando la sillabazione, il tracciato potrebbe essere molto più complesso, ma con un po' di pazienza sarebbe decifrabile come si è fatto per questo capoverso.

In realtà la frase che forma il capoverso che abbiamo tracciato è solo una mezza verità; per il tracciamento delle pagine è vera; per il tracciamento dei capoversi è falsa. Per avere il tracciamento completo come abbiamo mostrato sopra è necessario evitare di caricare il pacchetto *trace*, perché appunto configura i comandi di tracciamento in modo troppo ridotto per disporre di un buon tracciamento di pagine e di capoversi.

Chi scrive peraltro riconosce che in generale questi due tracciamenti, ricchissimi di informazioni, sono meno indispensabili di quelli che si ottengono con i comandi `\traceon` e `\traceoff`; chi scrive di solito riesce a capire i difetti di formazione dei capoversi e delle pagine senza ricorrere al loro tracciamento, ma leggendo solo la scarne informazioni associate agli avvisi di "Overfull" box verticali o orizzontali. Naturalmente questa è una questione di preferenze personali oltre che di cose che vengono scritte nei vari tipi di file sorgente quando provocano messaggi di errore o avvisi di cattiva composizione.

28.10 Conclusioni

Benché siano fortemente illuminanti, i comandi di tracciamento per le pagine e per i capoversi vengono usati raramente; i comandi `\traceon` e `\traceoff` vengono usati non tanto spesso, ma sono la vera ultima risorsa a cui accedere quando non si riesce a capire dai messaggi d'errore come eliminare gli errori che il programma di composizione si "ostina" a commettere. Inoltre è quanto mai opportuno leggere attentamente i messaggi d'errore; spesso la spiegazione dell'errore è descritta dal messaggio, ma con parole che non ci sono familiari o che non sappiamo comprendere. Solo la pratica insegna a capire questi messaggi, ma se non si comincia mai a usare gli strumenti descritti qui, non si riuscirà mai a capire il messaggi d'errore fino in fondo.

Nel TEXbook al capitolo 27, dedicato alla soluzione delle condizioni di errore, Knuth termina con queste due citazioni[2]:

Who can understand his errors?
— *Psalm 19:12* (c. 1000 B.C.)

It is one thing, to shew a Man that he is in an Error,
and another, to put him in possession of Truth.
— JOHN LOCKE, *An Essay Concerning Humane Understanding* (1690)

[2]Prima citazione: "Chi riesce a capire i propri errori?". Seconda citazione: "Una cosa è mostrare all'uomo che è in errore, e una differente metterlo in condizione di conoscere la verità."

Senza scendere troppo in considerazioni filosofiche, il problema dell'errore non solo era stato messo nero su bianco fin dai tempi in cui si ritiene che sia stata scritta la Bibbia, ma era anche un argomento dibattuto nel periodo pre-illuministico quando si evidenziava la differenza fra riconoscere l'errore rispetto a correggerlo.

Capitolo 29

Riepilogo della sintassi di LaTeX

In questo capitolo si riepiloga in modo molto succinto la sintassi di tutti (o quasi) i comandi LaTeX, anche quelli non descritti nei capitoli del testo. L'ordine del materiale segue l'appendice C del manuale di Leslie Lamport, il creatore di LaTeX. Non si tratta però di una semplice traduzione, anche se la materia sembra esposta nello stesso ordine; è una esposizione ragionata e commentata, anche alla luce di informazioni rese disponibili dopo la pubblicazione di quel manuale.

In particolare bisogna ricordare che recentemente molte parti del nucleo di LaTeX sono scritte inn linguaggio LaTeX 3, linguaggio che è ancora in via di sviluppo; a titolo di esempio cito il pacchetto *xparse* che fino a qualche anno fa era necessario caricare come interfaccia fra il linguaggio di LaTeX 2_ε e il nuovo linguaggio LaTeX 3. Recentemente le funzionalità di *xparse* sono incluse quasi tutte nel nucleo di LaTeX, e del pacchetto rimane solo il file di documentazione; questo è utilissimo perché con le sue funzionalità avanzatissime si possono fare cose incredibili. È bene che il lettore arrivato a questo punto della guida, dia una lettura attenta a questa documentazione e scoprirà un nuovo modo per creare le proprie macro.

Parentesi graffe e quadre Le parentesi graffe generalmente racchiudono o gli argomenti obbligatori dei comandi oppure un insieme di testo e di comandi e istruzioni che formano un gruppo; i comandi e le istruzioni mantengono la loro efficacia solo all'interno del gruppo, tranne alcuni comandi il cui effetto è globale; questi comandi globali sono:

`\newcounter`	`\pagenumbering`	`\newlength`	`\hyphenation`
`\setcounter`	`\thispagestyle`	`\newsavebox`	
`\addtocounter`	`\pagecolor`	`\newtheorem`	

Le parentesi quadre delimitano gli argomenti facoltativi di molti comandi; bisogna stare attenti alle possibili contraddizioni fra queste parentesi e le parentesi quadre che compaiono nel testo; queste potrebbero essere scambiate per quelle. Al fine di evitare queste incomprensioni è opportuno racchiudere le parentesi quadre testuali (o le frasi racchiuse fra parentesi quadre) fra parentesi graffe. I comandi con cui potrebbero succedere queste incomprensioni sono i seguenti:

`\\`	`\linebreak`	`\nolinebreak`	`\newcounter`	`\twocolumn`
`\item`	`\pagebreak`	`\nopagebreak`	`\newtheorem`	`\suppressfloats`

Si scriverà per esempio

```
\begin{itemize}
\item {[omissis]} il conte disse allora: "<Ah, ah!">
\item[{[omissis]}] serve per indicare
                   l'omissione di una parte di testo.
...
```

Si noti che con l'estensione fornita dal pacchetto *xparse* si possono definire con argomenti obbligatori e facoltativi racchiusi fra delimitatori diversi; quindi bisogna stare molto attenti alla documentazione dei singoli comandi. La regola appena enunciata per l'uso delle graffe e delle quadre vale però per tutti i comandi del nucleo originale di LaTeX.

Gli argomenti degli ambienti Quando si apre un ambiente, solo il comando di apertura può ricevere degli argomenti, mentre il comando di chiusura non ne può accettare alcuno. Se l'ambiente ha un nome che contiene un asterisco finale, questo va ripetuto anche nel comando di chiusura; l'asterisco non è un argomento, ma fa parte integrante del nome. Invece gli ambienti definiti tramite le funzionalità del pacchetto *xparse* permettono di usare gli argomenti obbligatori e facoltativi anche durante l'esecuzione dei comandi di chiusura.

Comandi fragili e robusti Quasi tutti i comandi di LaTeX sono robusti, nel senso che eseguono quel che devono eseguire senza che questa operazione possa essere modificata dalla posizione che essi hanno, per esempio negli argomenti di altri comandi. Sono fragili gli altri, e per evitare che possano combinare pasticci, bisogna proteggerli con `\protect` premesso al loro nome. Il LaTeX 3 Team sta facendo l'impossibile per eliminare e/o modificare i pochi comandi fragili che rimangono, tuttavia ce ne sono ancora alcuni.

I comandi fragili si possono comportare male quando sono contenuti fra gli argomenti mobili di altri comandi; sono argomenti mobili quelli che devono venire elaborati in fasi successive, per esempio quando devono essere scritti nei file ausiliari e poi riletti ed elaborati in una fase successiva quando si rilancia LaTeX.

I comandi con argomenti mobili sono tipicamente quelli che scrivono qualcosa che deve poi comparire negli indici generale e/o analitico; nei glossari, nelle liste delle figure e/o delle tabelle, eccetera; gli argomenti di `\footnote` e

di `\footnotetext`; gli argomenti di `\typein` e di `\typeout` che consentono di creare dei file `.tex` interattivi; il contenuto dell'ambiente *letter* definito all'interno della classe `letter`; l'argomento di `\thanks` usato tipicamente nei titoli, specialmente per indicare le afferenze degli autori; l'argomento facoltativo di `\bibitem`; l'argomento delle *@-espressioni* nella definizione degli incolonnamenti degli ambienti *tabular* e *array*.

È opportuno usare `\protect` con parsimonia, per non correre il rischio di inserirlo dove proprio non ci vuole; conviene inserirlo a posteriori quando l'esecuzione della compilazione mostra errori irrecuperabili, spesso di difficile diagnosi, ma altrettanto spesso dovuti alla fragilità di qualche comando.

I comandi e gli ambienti definiti con *xparse* sono robusti per costruzione. C'è chi ha fatto dei confronti fra la robustezza dei comandi nativi di TEX e i comandi corrispondenti di LATEX; prima degli interventi del LATEX Team era relativamente facile incontrare casi di fragilità, ma era risultato che la gestione dei contatori di TEXera robusta, mentre quella dei contatori di LATEX era fragile. La causa è risultata comprensibile dal fatto che i contatori LATEXformano una gerarchia, mentre quelli TEXsono indipendenti. Tuttavia sono cose a cui fare attenzione e i problemi di fragilità si possono sempre superare.

**Il comando ** Il comando \\ serve sia per andare a capo durante la composizione del testo normale, eventualmente contenuto dentro ambienti che specificano il tipo di composizione (testi in display), sia nella composizione di tabelle o matrici, quando si vuole terminare una riga di quelle strutture. I due usi possono entrare in collisione, specialmente nelle colonne definite con i descrittori di colonna `p`, `m` o `b`. Per evitare che un comando \\ usato con il testo che deve venire composto in una colonna `p`, `m` o `b` di una tabella possa venire male interpretato come comando relativo alla tabella, basta racchiudere l'intero contenuto della cella dentro un gruppo formato dalla coppia di parentesi graffe `{...}`; alternativamente si può usare il comando `\newline`.

Quando LATEX compone un capoverso si eviti di usare due comandi \\ di seguito; si usi invece l'argomento facoltativo per aggiungere altro spazio; la sintassi del comando, lo si ricorda, è:

```
\\[⟨spazio⟩]
\\*[⟨spazio⟩]
```

dove la versione asteriscata impedisce un salto pagina prima della nuova riga; a causa di queste diverse modalità di andare a capo, il comando \\ è fragile. Non più con la nuova definizione tramite LATEX 3.

Gli ambienti (anche nelle versioni asteriscate) e i comandi che accettano il comando \\, a parte il testo normale, sono i seguenti:

`\shortstack`	`\author`	`\title`	`\date`
eqnarray	*tabbing*	***array***	***tabular***

Gli ambienti *array* e *tabular* sono stati messi in evidenza perché, insieme con gli altri ambienti per comporre materiale tabulare, bisogna evitare che il segno di 'a capo' `\\` del testo possa essere frainteso per un comando di fine riga `\\` della tabella; questa possibile confusione è particolarmente insidiosa se la parte testuale viene composta con una delle dichiarazioni `\centering`, `\raggedright` o `\raggedleft`. Per evitare di confondere il significato di `\\`, si suggerisce di usare per l'"a capo' testuale il comando `\newline`.

Inoltre se si specifica uno ⟨*spazio*⟩ come argomento di `\\` all'interno di una tabella o di una matrice, e se questo è troppo piccolo, potrebbe non manifestarsi nessun effetto visibile, perché il piccolo spazio viene assorbito dalla profondità dello `\strut` (pilastrino) che compare in ogni riga delle tabelle e delle matrici.

29.1 La struttura del documento

```
\begin{filecontents}[⟨opzioni⟩]{⟨nome file⟩}
⟨contenuto del file⟩
\end{filecontents}
...
\documentclass[⟨lista di opzioni⟩]{⟨classe⟩}
⟨preambolo⟩
\begin{document}
⟨testo del documento⟩
\end{document}
```

La dichiarazione `\documentclass` eccezionalmente può essere preceduta da uno o più ambienti *filecontents*, ma spesso è meglio inserire questo ambiente nel corpo del preambolo del documento; essi servono, in particolar modo, quando si inviano i file sorgente ad altri collaboratori, per specificare i pacchetti non standard da cui dipende la composizione del documento. Servono per introdurre i metadati necessari per creare file archiviabili; servono per disporre del codice per comporre brani di testo da ripetere diverse volte. Questi file vengono estratti dai file sorgente la prima volta che il documento viene elaborato nella nuova situazione e memorizzati con il loro ⟨*nome file*⟩ nella cartella corrente, cosicché quando la compilazione comincia per davvero, LaTeX dispone di tutti i file di cui ha bisogno, compresi, appunto, i file non standard allegati al file sorgente. La presenza delle ⟨*opzioni*⟩ serve per evitare che nel file che si crea venga immesso un gruppo di righe commentate che permettono di configurare il file creato, o di consentirne la riscrittura, o di evitare la presenza di codice inutile o dannoso per il tipo particolare di contenuto del file, per esempio i file di che formano database bibliografici, i file di metadati, i file grafici in linguaggio PostScript, e simili.

29.2 Periodi e capoversi

29.2.1 Periodi

Apostrofi e virgolette

apostrofo: `'`
virgolette semplici rialzate: `‘`⟨*testo*⟩`’`
virgolette doppie rialzate: `"`⟨*testo*⟩`"`
virgolette caporali: `"<`⟨*testo*⟩`">` oppure con encoding `T1`: `<<`⟨*testo*⟩`>>`

Lineati

lineetta o trattino: `-`
trattino medio per intervalli numerici: `--`
tratto o lineato lungo per gli incisi e per i dialoghi: `---`

Spazi

spazio fine: `\,`
spazio interparola: `\␣`
spazio interparola che impedisce di andare a capo: `~`
spazio di fine periodo: `\@`

Il comando `\@` inserito dopo una lettera maiuscola e prima di un punto fermo serve per evitare che questo sia scambiato per un punto di abbreviazione. L'interprete, infatti, ritiene che un punto dopo una lettera minuscola sia un punto di fine periodo, mentre un punto dopo una lettera maiuscola indichi una abbreviazione. Si osservi la spaziatura quando si scrive M. Rossi e quando, invece, si scrive m. rossi. In questo secondo caso, lo spazio che segue m. dovrebbe essere più grande dello spazio che segue M. e comunque maggiore dello spazio interparola. Si confrontino ancora le due frasi:

Vado all'ufficio postale a pagare l'IMU. Poi torno a casa.
Vado all'ufficio postale a pagare l'IMU. Poi torno a casa.

Nel secondo caso *non* è stato inserito il comando `\@` dopo la parola IMU. La differenza è minima, ma un occhio esercitato la rileva. È per questo che nella composizione della bibliografia, dove le abbreviazioni sono numerose, viene usato di default il prossimo comando `\frenchspacing` che rende uguali tutti gli spazi dopo i segni di interpunzione, in modo che non ci sia da preoccuparsi se un punto indichi una abbreviazione o un punto fermo.
`\frenchspacing` serve per rendere uguali gli spazi dopo qualsiasi segno di interpunzione. Componendo in italiano con `pdflatex`, `babel-italian` non imposta `\frenchspacing` mentre con `lualatex` e `xelatex`, `gloss-italian` imposta di default questa spaziatura.
`\nonfrenchspacing` serve per aumentare gli spazi dopo i punti di interpunzione che marcano la fine della frase o del periodo, in particolare dopo

i comunissimi punti fermi. Secondo alcuni stimati book designer, la spaziatura alla francese sarebbe da evitare sempre; secondo altri dovrebbe essere l'unica ad essere usata. Si noti che questo intero documento è stato composto con la dichiarazione `\frenchspacing`.

Segni speciali

$	`\$`	%	`\%`	{	`\{`	_	`\_`
&	`\&`	#	`\#`	}	`\}`		

Loghi

`\LaTeX`	Logo di LaTeX
`\LaTeXe`	Logo di LaTeX 2_ε
`\TeX`	Logo di TeX

`\today` produce la data del giorno

`\emph{`⟨*testo*⟩**`}`** Evidenzia ⟨*testo*⟩ scrivendolo in corsivo all'interno di un contesto in tondo e in tondo all'interno di un contesto corsivo.

`\mbox{`⟨*testo*⟩**`}`** scrive ⟨*testo*⟩ all'interno di una scatola orizzontale in modo che non possa, per esempio, essere divisa in sillabe in fin di riga; questo comando può essere usato anche in un contesto matematico, ma di fatto, se non ci sono motivi speciali, in matematica si possono usare sia i comandi di scelta dei font da usare in un contesto testuale, per esempio `\textrm`, rinunciando però alla prerogativa delle matematica di scegliere il font della giusta dimensione, oppure i comandi `\text` oppure `\intertext` messi a disposizione dal pacchetto *amsmath*.

29.2.2 Capoversi

Un capoverso si può iniziare o terminare con uno dei seguenti comandi; in ogni caso una o più righe assolutamente vuote, senza neanche i segni di commento, servono per terminare un capoverso.

`\noindent` comincia un capoverso senza inserirne il tipico rientro.

`\indent` comincia un capoverso con il tipico rientro, anche quando normalmente esso non sarebbe inserito.

`\par` termina un capoverso e non è più necessario lasciare altre righe vuote.

I capoversi vengono composti secondo le seguenti giustezze:

`\textwidth` è la giustezza normale fissata dal file di classe; se la si vuole cambiare bisognerebbe farlo solo nel preambolo.

`\linewidth` è la giustezza corrente; può essere inferiore o maggiore del valore generale `\textwidth` a seconda dell'ambiente all'interno del quale si sta componendo.

`\columnwidth` è la giustezza di una colonna, quando si sta componendo su due o più colonne.

`\columnsep` è lo spazio di separazione fra due colonne adiacenti.

`\columnseprule` è lo spessore del filetto verticale che separa le colonne quando si compone su due o più colonne.

`\parindent` è l'ammontare del rientro del capoverso.

`\baselineskip` è lo scartamento stabilito con la scelta del corpo del font che si sta usando, eventualmente modificato dal comando seguente.

`\linespread{`⟨*fattore*⟩`}` modifica lo scartamento mediante il coefficiente moltiplicativo ⟨*fattore*⟩; per attivare questa specifica deve essere impartito il comando `\selectfont` in modo esplicito o implicito (è implicito se si specifica un altro comando di cambiamento di font).

`\lineskip` interlinea aggiuntiva da aggiungere fra due righe adiacenti per evitare che si sovrappongano.

`\lineskiplimit` valore minimo della distanza fra due righe successive, scendendo sotto il quale TEX introduce interlinea supplementare pari a `\lineskip`.

`\parskip` rappresenta uno spazio di interlinea (contrografismo) supplementare da inserire prima di ogni nuovo capoverso; generalmente questo spazio è nullo; talvolta è rappresentato da una *lunghezza elastica* (vedi più avanti nel paragrafo 29.8.2) dotata di un valore naturale nullo, ma di un piccolo allungamento, talvolta anche di un piccolo accorciamento, che consente di giustificare le pagine affacciate anche se contengono un numero di righe leggermente diverso. Questo normalmente succede quando, per esempio, una formula di una certa dimensione verticale o un nuovo paragrafo deve cominciare verso la fine di una pagina, ma se non c'è abbastanza posto, l'oggetto viene spostato alla pagina successiva. La pagina rimarrebbe 'mozza' e, per evitare questo inestetismo, il programma di composizione sfrutta l'elasticità di questi contrografismi per distribuire le righe della pagina in modo che le due pagine affacciate terminino allo stesso livello.

`\enlargethispage{`⟨*lunghezza*⟩`}` allunga la pagina (in realtà allunga l'altezza della griglia di stampa, `\textheight`) della ⟨*lunghezza*⟩ specificata; per ovvi motivi conviene che questa lunghezza sia specificata con un numero intero e piccolo (1 o 2) di righe di testo. Il comando asteriscato `\enlargethispage*{`⟨*lunghezza*⟩`}`, prima di allungare la griglia di stampa, cerca di comprimere i contrografismi verticali presenti nella pagina stessa al fine di farci stare più materiale; in generale è più conveniente usare il comando asteriscato. Se la ⟨*lunghezza*⟩ ha un valore negativo, la griglia viene accorciata.

29.2.3 Note in calce

`\footnote[`⟨*numero*⟩`]{`⟨*testo*⟩`}`

Se viene specificato il ⟨*numero*⟩ non vi si può fare riferimento in modo simbolico attraverso i comandi `\ref`, `\label`, eccetera. Il contatore `footnote` viene rappresentato con un numero a esponente, tranne che nelle note interne all'ambiente *minipage* dove le note sono numerate con lettere minuscole, sempre a esponente.

In certe circostanze è possibile separare il comando che inserisce il richiamo della nota dal comando che effettivamente la scrive:

```
\footnotemark[⟨numero⟩]
...
\footnotetext[⟨numero⟩]{⟨testo⟩}
```

Per i parametri stilistici di composizione si veda il prossimo paragrafo.

29.2.4 Note marginali

```
\marginpar[⟨nota a sinistra⟩]{⟨nota a destra⟩}
```

La ⟨*nota a sinistra*⟩ serve facoltativamente per specificare un testo con una modalità di composizione diverso da quello della ⟨*nota a destra*⟩.

I parametri stilistici con cui vengono composte le note in calce e quelle marginali sono le seguenti.

\footnotesep è una lunghezza che rappresenta l'altezza dello **\strut** messo all'inizio di ogni nota; aumentando questa altezza ogni nota viene staccata di più dalla precedente.

\footnoterule è il comando con il quale vengono collocati nella lista verticale di oggetti che compongono la pagina il filetto orizzontale e gli spazi adiacenti, in modo da marcare la separazione fra il testo e le note in calce; Complessivamente il filetto e gli spazi adiacenti devono avere una altezza totale nulla, quindi bisogna usare anche dello spazio 'negativo'; infatti la definizione di default di questo comando è la seguente:

```
\def\footnoterule{\kern-3\p@ \hrule \@width 2in \kern 2.6\p@}
```

Benché sia scritto con comandi nativi, si capisce che prima si arretra di 3 pt, poi si inserisce un filetto lungo due pollici e spesso 0,4 pt (questo è il valore di default), e infine si avanza di 2,6 pt: si constata che $-3\,\text{pt} + 0{,}4\,\text{pt} + 2{,}6\,\text{pt} = 0\,\text{pt}$. Si può ridefinire questo comando con **\renewcommand** ma la definizione deve sempre corrispondere ad una serie di oggetti la cui altezza complessiva sia nulla.

\marginparsep è la larghezza dello spazio che separa le note marginali dal testo a cui sono affiancate.

\marginparpush è la distanza minima fra due note marginali consecutive.

\marginparwidth è la giustezza con cui sono composte le note marginali; va da sé che il margine che le deve contenere deve essere sufficientemente largo e deve tenere conto anche di **\marginparsep** e di un buono spazio verso il taglio della pagina affinché dopo il taglio non si debba scoprire che anche parte della nota è stata tagliata.

\reversemarginpar è una istruzione booleana che imposta lo scambio del margine nel quale vengono composte le note marginali; agisce solo quando si

\‘{a}	à	\’{a}	á	\={a}	ā
\v{a}	ǎ	\^{a}	â	\"{a}	ä
\~{a}	ã	\u{a}	ă	\.{a}	ȧ
\H{a}	a̋	\t{oo}	o͡o	\c{t}	ţ
\d{a}	ạ	\b{a}	a̲	\r{u}	ů
\i	ı	\j	ȷ	\oe	œ
\OE	Œ	\ae	æ	\AE	Æ
\aa	å	\AA	Å	\o	ø
\O	Ø	\l	ł	\L	Ł
\ss	ß	?‘	¿	!‘	¡
\dag	†	\ddag	‡	\S	§
\P	¶	\copyright	©	\pounds	£

Tabella 29.1: I comandi per gli accenti testuali e i simboli speciali di molte lingue straniere

compone ad una sola colonna, perché a due colonne le note marginali della colonna di sinistra vengono composte comunque a sinistra, e quelle relative alla colonna di destra, a destra di questa.

\normalmarginpar ristabilisce la composizione delle note marginali nel margine esterno.

29.2.5 Accenti e simboli speciali

Premesso che con l'uso del pacchetto *inputenc* e della tastiera nazionale la necessità di ricorrere ai comandi per gli accenti è fortemente diminuita, tuttavia è utile saper gestire anche i comandi più insoliti al fine di comporre correttamente anche parole in lingue straniere che fanno uso di segni che non compaiono sulla tastiera nazionale.

I comandi per gli accenti testuali sono raccolti nella tabella 29.1, mentre quelli per il contesto matematico sono raccolti nella tabella 13.7.

29.3 Suddivisione del testo e indici

29.3.1 Comandi di sezionamento

Tutti i comandi di sezionamento propriamente detti, e gli altri comandi simili hanno una sintassi comune:

comando[⟨*titolo breve*⟩]{⟨*titolo lungo*⟩}

Essi sono:

Sezionamento	Livello	Macro che contiene il nome
`\part`	−1	`\partname`
`\chapter`	0	`\chaptername`
`\section`	1	
`\subsection`	2	
`\subsubsection`	3	
`\paragraph`	4	
`\subparagraph`	5	
`\caption`	—	`\figurename` `\tablename`

Tabella 29.2: Numeri e nomi associati al livello di sezionamento

```
\part            \chapter     \section        \subsection
\subsubsection   \paragraph   \subparagraph   \caption
```

Il comando `\chapter` non è definito per la classe *`article`*; nessun comando di sezionamento è definito per la classe *`letter`*. Il livello di sezionamento `\part` non è obbligatorio; tutti gli altri comandi devono essere contenuti sotto un comando del livello precedente.

Il comando `\caption` non è propriamente un comando di sezionamento, ma condivide con questi comandi diverse proprietà sintattiche e operative.

Le parole che eventualmente vengono scritte nelle intestazioni, come 'Part' o 'Parte', 'Chapter' o 'Chapître', vengono impostate tramite l'opportuna chiamata al pacchetto *`babel`* con le opzioni per la lingua del documento o *`polyglossia`* con le specificazioni delle lingue del documento. In ogni caso esse sono contenute nei comandi `\partname`, `\chaptername`, `\figurename`, `\tablename`, per cui possono venire ridefinite a piacere.

Ogni comando di sezionamento oltre a comporre l'intestazione della divisione specifica mediante il ⟨*titolo lungo*⟩, scrive negli opportuni file accessori il ⟨*titolo breve*⟩ da inserire nei corrispondenti indici e, con certi stili di pagina, nelle testatine.

Ogni divisione del documento può venire numerata e può comparire nell'indice; questi due fatti sono controllati dai valori numerici contenuti nei due contatori `secnumdepth` e `tocdepth`. Le sezioni sono numerate se il loro livello è associato ad un numero minore o uguale a quello contenuto in `secnumdepth`; i loro titoli vanno a finire nell'indice se il loro livello corrisponde ad un numero minore o uguale a quello contenuto in `tocdepth`; si veda la tabella 29.2.

Per produrre gli indici si usano i comandi:

```
\tableofcontents    \listoffigures    \listoftables
```

Le informazioni per produrre questi indici sono contenute in tre file ausiliari le cui estensioni sono rispettivamente `.toc`, `.lof` e `.lot`.

Le versioni asteriscate dei comandi di sezionamento producono solo l'intestazione non numerata e non producono niente nell'indice né nelle testatine. Il comando `\caption` non possiede la variante asteriscata.

Con alcune classi standard, come *`book`*, si possono considerare comandi di sezionamento anche `\frontmatter`, `\mainmatter`, `\backmatter`; non c'è dubbio che in un documento complesso come un libro queste tre parti costituenti il documento debbano essere distinte e i tre comandi suddetti eseguono questo sezionamento; apparentemente non scrivono nulla nel documento, ma queste tre dichiarazioni modificano in modo sostanziale il modo di comporre il documento stesso, in particolare il comportamento dei comandi di sezionamento.

1. Il comando `\frontmatter` imposta la numerazione delle pagine con numeri romani minuscoli; fa sì che i comandi di sezionamento *non asteriscati* non producano titoli numerati, ma i loro titoli vengono inseriti nell'indice.
2. Il comando `\mainmatter` reimposta lo stile di numerazione delle pagine in cifre arabe e consente che i comandi di sezionamento *non asteriscati* abbiano i loro titoli numerati ed eventualmente inseriti nelle testatine se lo stile delle pagine lo prevede e se i contatori `tocdepth` e `secnumdepth` lo consentono.
3. Il commando `\backmatter` non reimposta la numerazione delle pagine, quindi la numerazione in cifre arabe prosegue quella della parte principale del testo, ma di nuovo i comandi di sezionamento *non asteriscati* non sono più numerati ma possono avere i loro titoli nelle testatine, come nella parte iniziale. A questo proposito ci sono alcuni che ritengono che le appendici facciano parte della parte finale (quindi vi appaiano solo con i loro titoli senza la consueta numerazione letterale), e altri ritengano che, quando le appendici sono più di una, esse debbano essere numerate (con la consueta numerazione letterale) e debbano apparire nell'indice, e perciò debbano far parte della parte principale.

La descrizione delle caratteristiche di questi speciali comandi di sezionamento implica che se si usano i comandi di sezionamento *asteriscati*, `\chapter*`, `\section*`, eccetera, si comportano in modo diverso nel senso che producono sezioni non numerate i cui titoli non vanno a finire nelle testatine. Quando si usano classi che non prevedono l'uso di `\frontmatter` e compagni, ci sono due modi per aggirare questi due "scogli":

1. Si possono impostare i contatori `secnumdepth` e `tocdepth` in modo da replicare il funzionamento di `\frontmatter`:

   ```
   \setcounter{secnumdepth}{-2}
   \setcounter{tocdepth}{2}
   ```

   In questo modo *non* bisogna usare i comandi asteriscati, ma bisogna usare i comandi normali *non asteriscati*.

2. Si possono usare i comandi asteriscati, ma bisogna fornire con ulteriori comandi le informazioni che servono; per esempio con `\chapter*` si può fare così:

   ```
   \chapter*{⟨Titolo del capitolo non numerato⟩}
   \addcontentsline{toc}{chapter}{⟨Titolo del capitolo non numerato⟩}
   \markboth{⟨Titolo del capitolo non numerato⟩}{⟨Titolo del capitolo non numerato⟩}
   ```

   Per i dettagli si veda quanto scritto qui di seguito a proposito del comando `\addcontentsline`; questo comando serve per scrivere il suo contenuto nell'indice con lo stile delle righe riferite ai capitoli; se non interessa inserire la voce nell'indice, non si usa l'intera riga.
   Invece il fatto che `\markboth` contenga lo stesso titolo sia per le testatine di sinistra sia per quelle di destra, nasce dalla considerazione che un capitolo non numerato difficilmente verrebbe diviso in sezioni; se lo fosse, anche queste dovrebbero essere ottenute con comandi asteriscati e dovrebbero venire trattate in modo simile a quello dei capitoli:

   ```
   \section*{⟨Titolo del paragrafo non numerato⟩}
   \addcontentsline{toc}{section}{⟨Titolo del paragrafo non numerato⟩}
   \markright{⟨Titolo del paragrafo non numerato⟩}
   ```

   In questo modo eventuali sezionamenti del capitolo non numerato avrebbero nelle testatine di destra il titolo del paragrafo non numerato, ma resterebbero inalterate le testatine di sinistra.

Il comando `\addcontentsline` ha la seguente sintassi:

```
\addcontentsline{⟨estensione⟩}{⟨livello⟩}{⟨voce⟩}
```

dove ⟨*estensione*⟩ è l'estensione del file ausiliario per l'indice specifico; ⟨*livello*⟩ è il nome (non il numero) del livello di sezionamento, espresso dallo stesso nome del comando, senza però il backslash; la ⟨*voce*⟩ è quanto viene scritto nel file indice. Generalmente quanto viene scritto nell'indice è costituito dall'espressione

```
\protect\numberline{⟨numero⟩}{⟨titolo⟩}
```

dove ⟨*numero*⟩ è il numero della sezione, per esempio 3.8 per l'ottavo paragrafo del capitolo tre; va da sé che una sezione non numerata ha questo campo vuoto. Il ⟨*titolo*⟩ non è altro che il titolo (breve) della sezione.

Per le sezioni non numerate lasciare vuoto il campo riservato al ⟨*numero*⟩ implica che il ⟨*titolo*⟩ viene incolonnato con gli altri titoli delle sezioni numerate; se invece si omette tutta la parte `\protect\numberline{`⟨*numero*⟩`}`, allora il

⟨*titolo*⟩ viene allineato al margine sinistro. Si veda, per esempio, nell'indice generale di questo testo dalla voce 'Bibliografia' all'ultimo indice analitico.

Per l'indice analitico e il glossario, i vari elementi vengono raccolti mediante i rispettivi comandi `\index` e `\glossary`, ma il contenuto del loro argomento varia molto a seconda di come si intende usare l'informazione raccolta. In ogni caso nessuna informazione viene raccolta se nel preambolo mancano i comandi `\makeindex` e, rispettivamente, `\makeglossary`; l'indice analitico e/o il glossario non vengono composti se non si dà esplicitamente il comando di comporli, di solito attraverso comandi messi a disposizione da pacchetti esterni. Altrimenti bisogna usare gli ambienti *index* e *glossary* specificando a mano il nome dei file che contengono l'indice o il glossario elaborati adeguatamente.

29.4 Classi, pacchetti e stili delle pagine

29.4.1 Classe del documento

La scelta della classe va eseguita con il comando:

`\documentclass[`⟨*opzioni*⟩`]{`⟨*classe*⟩`}`

Le classi standard sono *book*, *report*, *article*, *letter*, *slides*, *minimal*, *proc*, *ltxdoc*; queste sono affiancate dalle numerosissime altre classi non standard che popolano gli archivi internazionali e nelle installazioni complete del sistema TEX.

Le opzioni standard sono le seguenti:

10pt, definisce il corpo di default per il documento; però può venire trasmessa anche ad altri pacchetti, fra i quali *type1ec* il quale con questa opzione ricava i disegni di tutti i caratteri dai font in corpo 10 pt, ingrandendoli o rimpicciolendoli a seconda delle richieste del compositore. Questo modo di procedere è accettabile solo quando si sta lavorando con font vettoriali. Il file di uscita risulta meno ingombrante, la qualità peggiora un poco, ma questa riduzione è visibile ad occhio nudo. Tra l'altro si noti la differenza di resa dei font ingranditi o rimpiccioliti; qui si mostra lo stesso testo riportato al corpo 10 partendo da un font di corpo 5 e da un font di corpo 17: partendo da un font di corpo 5 e partendo da un font di corpo 17. L'effetto opposto si otterrebbe partendo da un font di corpo 10 riducendolo al corpo 5 o ingrandendolo al corpo 17.

11pt, *12pt*, servono per scegliere il corpo da 11 pt o da 12 pt come corpo normale.

letterpaper, *legalpaper*, *executivepaper*, *a4paper*, *a5paper*, *b5paper*, servono per scegliere il supporto della carta secondo i formati specificati con le sigle suddette, precisamente:

letter	8,5 in × 11 in	A4	210 mm × 297 mm
legal	8,5 in × 14 in	A5	148 mm × 210 mm
executive	7,25 in × 10,5 in	B5	176 mm × 250 mm

Siccome l'opzione di default è *letterpaper* per scrivere in Europa è sempre necessario specificare *a4paper*.
Le dimensioni così specificate vengono passate anche al file `.pdf` nel caso che si usi `pdflatex` o uno degli altri programmi che producono l'uscita in formato PDF, in quanto questo formato ha bisogno di conoscere le dimensioni del supporto di uscita. Il file `.dvi` a stretto rigore non ne ha bisogno, ma poi diventa necessario se questo tipo di file viene convertito nei file `.ps` oppure `.pdf`.

landscape, serve per scambiare fra di loro la larghezza e l'altezza del supporto.

final, *draft*, servono per specificare se si sta componendo la versione finale, oppure una bozza; quando si compone una bozza, vengono evidenziate le righe fuori giustezza con un vistoso rettangolo nero e non vengono importate le figure, ma al loro posto viene segnato solo un rettangolo delle giuste proporzioni con il nome del file grafico scritto dentro.

oneside, *twoside*, servono per specificare se si vuole comporre solo sul recto o anche sul verso della carta.

openright, *openany*, servono per specificare se i capitoli possono iniziare solo sulle pagine di destra oppure su qualunque pagina.

onecolumn, *twocolumn*, servono per specificare se si vuol comporre su un'unica colonna o su due colonne.

notitlepage, *titlepage*, servono per specificare se il titolo del documento va in testa alla prima pagina, nella quale comincia subito anche il testo, oppure se si deve riservare una pagina solo per il titolo. Di default la classe ***book*** è impostata con*titlepage*, mentre le classi ***report*** e ***article*** sono impostate con *notitlepage*.

openbib, serve per comporre la bibliografia (nelle classi che la riconoscono) con uno stile più aperto; generalmente le varie parti del riferimento bibliografico sono trattate come dei brevi capoversi senza rientro; però le loro righe dopo la prima risultano composte con un rientro, specificato mediante il parametro `\bibindent`, rispetto al margine sinistro.

leqno, serve per specificare che si desidera numerare le equazioni con il numero collocato a filo del margine sinistro della gabbia del testo. Se anche le equazioni vengono allineate a sinistra, potrebbe non esserci abbastanza spazio.

fleqn, serve per specificare che non si desiderano le equazioni centrate, ma si desidera che siano rientrate di una quantità fissa dal margine sinistro; questa quantità viene specificata mediante `\mathindent` che di default la imposta al valore della rientranza del margine sinistro delle liste di primo livello. Forse è un po' poco, specialmente se i numeri identificativi delle equazioni (che possono diventare relativamente lunghi quando esse sono numerose) con l'opzione precedente è messo accostato al margine sinistro dello specchio di stampa; si può sempre modificare `\mathindent` con il comando `\setlength`. Va notato, però, che con il valore di default di `\mathindent` non vengono introdotti altri valori di rientranza dei margini, e questo tipograficamente va bene, ma con numerose equazioni potrebbe

verificarsi l'interferenza dell'etichetta dell'equazione con l'equazione stessa. Il pacchetto *amsmath* evita sempre questa interferenza, perciò direi che sia meglio usare *amsmath* piuttosto di modificare `\mathindent`.

29.4.2 Pacchetti

La scelta del pacchetto si esegue con il comando `\usepackage` con la sintassi

`\usepackage[`⟨*opzioni*⟩`]{`⟨*pacchetto*⟩`}[`⟨*data*⟩`]`

dove ⟨*data*⟩ è una data in forma ISO (*aaaa/mm/gg* oppure *aaaa-mm-gg*) che rappresenta il limite inferiore della data corrispondente alla versione che si vuole usare; da quando esiste LaTeX 2ε (1994) i vari pacchetti sono stati aggiornati così che versioni successive non solo hanno (presumibilmente) meno errori o 'features' degli stessi pacchetti nelle loro versioni precedenti, ma anche funzionalità maggiori. Ecco che specificando la data si può essere sicuri di usare una versione abbastanza recente che presenta le funzionalità desiderate. I pacchetti standard (quelli curati dal LaTeX Project Team) sono:

alltt serve per definire l'ambiente *alltt* dove quasi tutti i caratteri speciali perdono il loro significato specifico e vengono usati come caratteri 'normali'.

amsmath costituisce la principale estensione per gli ambienti matematici; vedi più avanti l'uso dei pacchetti *amssymb* e *amsfonts*.

babel consente di comporre in diverse lingue; attenzione: *babel* sceglie le specifiche della lingua, compresa la sillabazione, ma non è *babel* che carica la sillabazione; se i pattern per la cesura di una data lingua non sono stati caricati durante la creazione del file di formato, di default *babel* usa i pattern per l'inglese, ottenendo cesure assolutamente incompatibili: per esempio, con l'italiano produrrebbe 'equ-azi-one' o 'dis-creto'; vedi il capitolo 25. Per pdflatex, ma anche per lualatex e xelatex

polyglossia consente di gestire le lingue per i documenti composti con lualatex o xelatex; come *babel*, specifica per ogni lingua anche la sua sillabazione, ma non ne carica i pattern se si usa xelatex perché sono già stati caricati tutti nel momento in cui si è creato il file di formato; invece con lualatex carica al momento solo i pattern per le lingue effettivamente usate nel documento. Solo per lualatex e xelatex

color serve per gestire i colori dei font e degli sfondi.

graphics serve per gestire le manipolazioni grafiche. Benché sia il pacchetto che svolge il lavoro richiesto, per l'utente è più comodo riferirsi al pacchetto seguente.

graphicx gestisce le operazioni grafiche e gli effetti speciali con una interfaccia utente più efficace; in realtà *graphicx* si occupa di caricare *graphics* e di passargli i comandi tradotti nella sua sintassi specifica.

ifthen permette di eseguire un fine controllo logico con una interfaccia utente più semplice di quella che offrono i comandi primitivi di TeX. Si veda più avanti la spiegazione del perché sia meglio usare altri pacchetti.

latexsym consente di usare i simboli speciali di LaTeX 2.09; questo pacchetto è disponibile solo per compatibilità con il passato; tutti quei simboli sono accessibili anche con il pacchetto *amssymb* che consente l'accesso ai font della American Mathematical Society descritti nel pacchetto *amsfonts*.

makeidx serve per gestire comodamente la creazione dell'indice analitico. Si possono usare in alternativa i pacchetti *imakeidx* o *indextools* che gestiscono anche la produzione di indici analitici multipli e consentono di sfruttare la proprietà dei sistemi moderni di lanciare comandi di sistema cosicché gli indici analitici possono essere prodotti direttamente con una sola esecuzione dei programmi di composizione.

pict2e estende le capacità grafiche dell'ambiente *picture* di LaTeX. Queste estensioni sono disponibili in modo stabile solo dal 2004; negli anni successivi gli sono state ulteriormente estese le funzionalità; si vede dunque come l'opzione della ⟨*data*⟩ non sia una cosa tanto insolita da usare. Le ultime modifiche sono riportate nelle *news letters* pubblicate circa ogni sei mesi, comunque con ogni aggiornamento annuale di TeX Live; si veda il capitolo 30.

showidx consente di scrivere le voci raccolte con `\index` direttamente sulla pagina dove sono state trovate; durante la lavorazione di un documento questa opzione si rivela utile per selezionare e confrontare le varie voci, sia per ridurre i duplicati sia per scegliere i riferimenti più adatti.

29.4.3 Stili delle pagine

Ogni classe inizializza lo stile delle pagine normali secondo le prescrizioni stilistiche generali della classe stessa. Il comando per scegliere lo stile è

```
\pagestyle{⟨stile⟩}
```

Il comando

```
\thispagestyle{⟨stile⟩}
```

imposta lo stile solo per la pagina corrente. Gli stili predefiniti nelle classi standard sono i seguenti.

`plain` contiene solo il piedino con il numero della pagina.

`empty` sia il piedino sia la testatina sono vuoti.

`headings` il piedino è vuoto ma la testatina è elaborata e automaticamente contiene sul verso delle pagine scritte con l'opzione *twoside* il titolo (breve) del capitolo e nella testatina del recto il titolo (breve) del paragrafo attivo all'inizio della pagina.

Questi titoli sono forniti dai comandi di sezionamento in modo automatico; se si desiderano testatine specifiche, che contengano elementi diversi o titolini correnti diversi bisogna usare i comandi `\markright` e `\markboth`, oppure bisogna servirsi di pacchetti di estensione come, per esempio, *fancyhdr*. Oppure bisogna servirsi di classi diverse da quelle standard.

`myheadings` in senso generale è simile allo stile `headings`, salvo che le informazioni da inserire nelle testatine debbono essere fornite esplicitamente dal compositore mediante i comandi `\markright` e `\markboth`.

La sintassi di questi comandi è la seguente:

```
\markright{⟨titolino di destra⟩}
\markboth{⟨titolino di sinistra⟩}{⟨titolino di destra⟩}
```

La numerazione delle pagine viene specificata con il comando

```
\pagenumbering{⟨stile di numerazione⟩}
```

dove gli stili di numerazione disponibili sono i seguenti:

`arabic` serve per la normale numerazione con cifre arabe.
`roman` serve per la numerazione romana in lettere minuscole.
`Roman` serve per la numerazione romana in lettere maiuscole
`alph` serve per la numerazione alfabetica in lettere minuscole.
`Alph` serve per la numerazione alfabetica in lettere maiuscole.
`fnsymbol` serve per rappresentare i numeri con la stessa sequenza di simboli usata per le note, per esempio l'asterisco, la spada, la doppia spada e via di seguito. I simboli in totale sono nove, quindi non è possibile numerare più di 9 cose. Generalmente questo tipo di numerazione non viene usato per le pagine, ma lo si usa abbastanza spesso per le note, in particolare per quelle del frontespizio.

Difficilmente si useranno numerazioni alfabetiche perché il numero totale delle pagine da numerare potrebbe superare 26, ma per le pagine le numerazioni arabe e romane vengono usate spesso. In Europa le cifre romane minuscole di solito sono rese con il carattere maiuscoletto, non con il tondo normale; per quel che riguarda l'italiano né *babel* né *polyglossia* forniscono le necessarie definizioni per usare il maiuscoletto.

Lo stile della pagina è vistosamente differente se si compone su una sola colonna o su due o più colonne; le classi standard consentono al massimo due colonne, ma mediante il pacchetto *multicol* non è difficile aumentare il numero delle colonne.

Quando però si usano le classi standard e si specifica di comporre ad una sola colonna, è possibile comporre alcune pagine su due colonne, consentendo di comporre anche una intestazione ad una colonna. Al contrario se si è specificata l'opzione per comporre su due colonne, allora è possibile comporre parte delle pagine su una sola colonna. I comandi da usare sono:

```
\twocolumn[⟨intestazione ad una colonna⟩]
\onecolumn
```

Entrambi i comandi iniziano sempre una nuova pagina poi cominciano a comporre come dice il loro nome.

Ovviamente lo stile della pagina dipende anche dalla giustezza e dalla sua relazione con il formato della carta, e quindi anche dei margini. Le giustezze orizzontali e verticali sono specificate mediante le lunghezze `\textwidth` e `\textheight` e i margini sono specificati mediante le lunghezze `\oddsidemargin` per le pagine di destra, `\evensidemargin` per le pagine di sinistra, `\topmargin` per il margine superiore; i margini laterali sono comunque i margini sinistri, e le parole 'odd' e 'even' (dispari o pari) si riferiscono alla posizione della pagina in relazione alla sua numerazione, sapendo che il verso di tutte le pagine è sempre pari e il recto sempre dispari. Questi margini sono riferiti allo spigolo superiore sinistro del foglio di carta e tutti sono 'un pollice' di meno di quello che ci si aspetterebbe; questo dipende dal fatto che inizialmente era previsto che tutti i driver di uscita avrebbero inserito di default un pollice di margine.

Non è il caso di mettersi a giocare con questi margini; se proprio si deve modificare l'aspetto geometrico di una pagina, è meglio usare una classe che consenta di farlo, oppure il pacchetto *geometry*.

29.4.4 Il frontespizio

È vero che se uno deve comporre un frontespizio in modo decoroso, è opportuno che studi il layout e la scelta del corpo e dello stile dei font in modo professionale. Tuttavia nei rapporti e negli articoli anche la modesta composizione di default operata dalla classi standard di LaTeX può essere accettabile.

Bisogna ovviamente specificare l'opzione *titlepage*; ma bisogna anche specificare nel preambolo un certo numero di informazioni.

`\title{`⟨*titolo*⟩`}` serve per specificare il titolo del documento; se questo titolo è sufficientemente lungo lo si può comporre su più righe inserendo il comando `\\`. Bisogna però ricordarsi di non andare a capo fra un articolo e il nome, fra una preposizione e il suo complemento, fra avverbi brevi, come 'non', e quanto segue; fa parte dell'eleganza della composizione. Riassumendo: nei titoli non si deve mai andare a capo dopo le congiunzioni, gli articoli, le preposizioni e i brevi avverbi; per evitarlo si deve usare il comando di legatura costituito dalla tilde `~`.

`\author{`⟨*nomi*⟩`}` serve per specificare i nomi degli autori; ma se gli autori sono più di uno bisogna separarli con `\and`; si può andare a capo usando `\\`. **Attenzione**: per ciascun autore sempre il o i nomi propri prima del o dei cognomi!

`\date{`⟨*testo*⟩`}` il testo di questo comando potrebbe essere una data, come suggerisce il nome del comando, ma potrebbe anche essere il nome di una conferenza, un luogo, qualunque informazione che permetta di identificare il documento oltre al suo titolo e ai suoi autori (i quali potrebbero aver prodotto un articolo con lo stesso titolo ad un altro convegno...).

`\thanks{`⟨*testo*⟩`}` questo comando può essere appeso sia ai nomi degli autori, per esempio per specificarne l'afferenza, sia al titolo, per esempio per associargli il nome dell'ente finanziatore delle ricerca, sia al testo specificato con `\date`. Esso funziona come il comando per inserire delle note a piè di pagina, solo che il riferimento di queste note non è né un numero né una lettera, ma un generico simbolo tratto da una lista che contiene l'asterisco, la spada, la spada doppia, eccetera.

Il tutto viene eseguito dando il comando `\maketitle`. Alternativamente e con risultati esteticamente variabili a seconda del gusto del compositore, si può comporre il frontespizio aprendo l'ambiente *titlepage* e scrivendoci dentro quello che si desidera, dove lo si desidera, con i font che si desiderano; talvolta la cosa produce buoni risultati, talaltra è meglio affidarsi alla composizione di default.

Nei rapporti e negli articoli spesso il frontespizio può contenere anche un breve riassunto; in LaTeX questo viene composto mediante l'ambiente *abstract*. Il riassunto può comparire come prima cosa dell'articolo e se questo è composto su due colonne, il sunto si trova nella prima colonna, subito sotto il titolo e le informazioni composte con `\maketitle`.

Ovviamente classi diverse da quelle standard possono mettere a disposizione del compositore degli ambienti preconfezionati di *titlepage* che dispongono le informazioni desiderate in modo diverso, più elegante, includendo alche il retro del frontespizio, con le informazioni legali, il copyright, gli avvisi di copia riservata o proibita, altre informazioni sulle persone che hanno collaborato a impaginare il testo, a fare i disegni, le fotografie, e simili. Alcune classi consentono anche di inserire nel retro del frontespizio anche il barcode classico a 10 cifre, sia quello moderno a 13 cifre. Bisogna però documentarsi accuratamente in merito alle classi alternative per scegliere quella che fa al caso proprio.

29.5 Testi in display

29.5.1 Citazioni e poesie

Gli ambienti per i testi in display sono:

```
\begin{quote} ⟨testo⟩ \end{quote}
\begin{quotation} ⟨testo⟩ \end{quotation}
\begin{verse} ⟨testo⟩ \end{verse}
```

29.5.2 Liste

Gli ambienti predefiniti per le liste sono:

```
\begin{itemize}
\item[⟨contrassegno⟩] ⟨testo⟩
...
\end{itemize}
```

```
\begin{enumerate}
\item[⟨contrassegno⟩] ⟨testo⟩
...
\end{enumerate}
```

```
\begin{description}
\item[⟨voce da descrivere⟩] ⟨testo di descrizione⟩
...
\end{description}
```

Per i primi due ambienti il ⟨*contrassegno*⟩ è davvero facoltativo; per l'ambiente ***description***, benché appaia come argomento facoltativo, in realtà esso è obbligatorio, perché non avrebbe senso dare una descrizione di nulla. Tuttavia se non si specifica nulla, nemmeno le parentesi quadre, si comincia una nuova descrizione senza etichetta che può essere vista come un nuovo capoverso. Non sembra una buona idea sfruttare così male le potenzialità di LaTeX.

L'ambiente principale, con il quale sono costruite tutte le altre liste, però, è l'ambiente ***list***, che è completamente configurabile in ogni dettaglio. La sintassi è:

```
\begin{list}{⟨contrassegno di default⟩}{⟨dichiarazioni⟩}
\item[⟨contrassegno personalizzato⟩] ⟨testo⟩
...
\end{list}
```

Il ⟨*contrassegno di default*⟩ è il modo di comporre il contrassegno che ***list*** produce quando non viene esplicitato un ⟨*contrassegno personalizzato*⟩ con il comando `\item`. Le ⟨*dichiarazioni*⟩, invece, sono una serie di istruzioni che specificano il modo di comporre eseguito da ***list***. Ci sono numerose possibilità che è meglio descrivere una alla volta.

`\topsep` è la distanza che separa il testo che precede la lista dalla prima voce della lista. La stessa distanza viene posta dopo la chiusura della lista.

`\partopsep` è lo spazio aggiuntivo aggiunto prima e dopo la lista se questa comincia un nuovo capoverso, cioè se nel file sorgente una riga completamente vuota precede il comando di apertura dell'ambiente.

`\itemsep` è lo spazio aggiuntivo che viene messo prima di una voce della lista se questa è preceduta da una riga bianca.

`\parsep` Siccome dentro le liste i capoversi solitamente non vengono rientrati, allora si può usare un piccolo spazio aggiuntivo fra due capoversi appartenenti alla stessa voce; questo spazio di separazione fra i *paragraphs* si chiama appunto `\parsep`.

`\leftmargin` è la rientranza di ogni voce della lista rispetto al testo circostante; liste annidate hanno rientranze sempre maggiori; il file di classe specifica le varie successive rientranze mediante i comandi `\leftmargini`, `\leftmarginii`, `\leftmarginiii` e `\leftmarginiv`.

`\rightmargin` è la distanza orizzontale fra il margine destro della lista corrente e il margine destro del testo che contiene la lista.

`\listparindent` è il rientro extra aggiunto (o anche tolto se ha un valore negativo) alla rientranza di ogni linea di ciascuna voce, tranne la prima riga; tutte le classi standard hanno questo parametro impostato a zero, ma non è impossibile che in altre classi o in liste personalizzate non si possa usare questo parametro.

`\itemindent` è la rientranza della prima riga di ogni voce della lista. Può anche avere un valore negativo.

`\labelsep` è la distanza minima che separa il ⟨*contrassegno*⟩ dal resto del testo nella prima riga di ogni voce.

`\labelwidth` è la larghezza prevista per il ⟨*contrassegno*⟩; a seconda della personalizzazione se questo contrassegno fosse più largo o più stretto di questa larghezza, il contrassegno viene fatto sporgere a sinistra, oppure viene fatto sporgere a destra ma cominciando il testo della voce più a destra in modo da mantenere costante lo spazio fra la fine del ⟨*contrassegno*⟩ e l'inizio della voce.

`\makelabel{`⟨*contrassegno*⟩`}` è la macro che effettivamente compone il contrassegno.

`\usecounter{`⟨*contatore*⟩`}` specifica che il ⟨*contatore*⟩ è quello usato per numerare gli elementi della lista e lo rende usabile con i comandi `\label` e `\ref`.

Vale la pena di provare a leggere la definizione dell'ambiente *enumerate* per capire meglio come si usi l'ambiente *list*. L'esempio che si porta è ripreso dalle definizioni, sparse in diversi file, per l'ambiente *enumerate* della classe *book* quando si compone con il corpo di default di 10 pt. Le definizioni vere e proprie dell'ambiente sono eseguite con comandi di basso livello e/o con comandi di servizio di LaTeX, ma si ritiene che siano abbastanza chiare da non richiedere una spiegazione, mentre se ne daranno per altri aspetti delle definizioni.

```
%                                        Definizioni in latex.ltx
\newcount\@enumdepth \@enumdepth = 0
\@definecounter{enumi}
\@definecounter{enumii}
\@definecounter{enumiii}
\@definecounter{enumiv}
%                                         Definizioni in book.cls
\renewcommand\theenumi{\@arabic\c@enumi}
\renewcommand\theenumii{\@alph\c@enumii}
\renewcommand\theenumiii{\@roman\c@enumiii}
\renewcommand\theenumiv{\@Alph\c@enumiv}
```

```
\newcommand\labelenumi{\theenumi.}
\newcommand\labelenumii{(\theenumii)}
\newcommand\labelenumiii{\theenumiii.}
\newcommand\labelenumiv{\theenumiv.}
\renewcommand\p@enumii{\theenumi}
\renewcommand\p@enumiii{\theenumi(\theenumii)}
\renewcommand\p@enumiv{\p@enumiii\theenumiii}
%                                        Definizioni in bk10.clo
\def\@listi{\leftmargin\leftmargini
            \parsep 4\p@ \@plus2\p@ \@minus\p@
            \topsep 8\p@ \@plus2\p@ \@minus4\p@
            \itemsep4\p@ \@plus2\p@ \@minus\p@}
\let\@listI\@listi
\@listi
\def\@listii {\leftmargin\leftmarginii
              \labelwidth\leftmarginii
              \advance\labelwidth-\labelsep
              \topsep    4\p@ \@plus2\p@ \@minus\p@
              \parsep    2\p@ \@plus\p@  \@minus\p@
              \itemsep   \parsep}
\def\@listiii{\leftmargin\leftmarginiii
              \labelwidth\leftmarginiii
              \advance\labelwidth-\labelsep
              \topsep    2\p@ \@plus\p@\@minus\p@
              \parsep    \z@
              \partopsep \p@ \@plus\z@ \@minus\p@
              \itemsep   \topsep}
\def\@listiv {\leftmargin\leftmarginiv
              \labelwidth\leftmarginiv
              \advance\labelwidth-\labelsep}
\def\@listv  {\leftmargin\leftmarginv
              \labelwidth\leftmarginv
              \advance\labelwidth-\labelsep}
\def\@listvi {\leftmargin\leftmarginvi
              \labelwidth\leftmarginvi
              \advance\labelwidth-\labelsep}
%                                       Definizioni in latex.ltx
\def\enumerate{%
  \ifnum \@enumdepth >\thr@@\@toodeep\else
    \advance\@enumdepth\@ne
    \edef\@enumctr{enum\romannumeral\the\@enumdepth}%
      \expandafter
      \list
        \csname label\@enumctr\endcsname
        {\usecounter\@enumctr\def\makelabel##1{\hss\llap{##1}}}%
```

```
  \fi}
\let\endenumerate =\endlist
```

Inizialmente si definiscono il contatore di livello e i quattro contatori per ciascuno dei quattro livelli; si ricorderà infatti che le liste, anche le enumerazioni, possono essere annidate l'una nell'altra, ma solo fino al quarto livello.

Successivamente nella classe *book* si definiscono i modi di rappresentare i vari contatori di enumerazione; il primo contatore viene stampato mediante `\@arabic\c@enumi`, cioè in cifre arabe; nella lista, però, esso viene marcato come il numero seguito da un punto: `\theenumi.` Per il primo livello basta così, ma per i livelli successivi ci vuole un prefisso; quando si cita il terzo elemento della seconda lista annidata, per esempio, non basta la scrittura del valore del secondo contatore, che viene stampato con una lettera mediante `\@alph\c@enumii`, ma nella lista questa lettera è racchiusa fra parentesi tonde `(\theenumii)`. Nella citazione del terzo elemento non basta che appaia 'c' oppure '(c)', ma ci vuole anche un prefisso che faccia riferimento alla prima lista; ecco quindi il prefisso `\p@theenumii` che fa si che quell'elemento sia citato come '2(c)'. Lo stesso avviene per gli elementi delle liste interne.

Nel file `bk10.clo`, che contiene le dimensioni e i corpi specificati per quando si compone con l'opzione di default *10pt*, sono definiti alcuni dei parametri descritti per l'ambiente *list*. Per l'enumerazione di primo livello il tutto è contenuto nella definizione del comando `\@listi`; questi spazi si riferiscono a tutte le liste di primo livello, non solo alle enumerazioni; precisamente sono definiti `\parsep`, `\topsep` e `\itemsep`; sono tutte lunghezze elastiche, perché contengono la componente di allungamento, introdotta da `\@plus`, e la componente di accorciamento, introdotta da `\@minus`; il comando `\p@` indica una lunghezza predefinita del valore di un punto tipografico. Fra le definizioni contenute in `\@listi` compare anche `\leftmargin`, a cui viene assegnato il valore di `\leftmargini`; questo a sua volta è definito nella classe *book* come una quantità espressa in unità di misura `em`, legate, cioè, al corpo del font in uso in quel contesto (potrebbe anche essere una nota o una citazione scritta in corpo minore). Definizioni analoghe appaiono anche per i livelli interni di annidamento.

Infine viene la definizione vera e propria che ricorre all'ambiente *list* così come è contenuta in `latex.ltx`. In questa definizione per prima cosa si verifica che il livello di annidamento già raggiunto non sia superiore a tre; se così fosse si emette un messaggio di errore mediante il comando `\@toodeep`, altrimenti si incrementa il contatore del livello di annidamento e si procede alle varie definizioni; si noti che per fare riferimento ai parametri corrispondenti al livello di annidamento corrente si fa riferimento al suo nome ottenuto agglutinando il prefisso `enum` con il valore del contatore di annidamento espresso in numeri romani minuscoli; questo risultato si ottiene con i comandi primitivi `\csname` e `\endcsname`; in generale, l'utente che deve usare l'ambiente *list* per confezionare una lista speciale non ha bisogno di ricorrere a questi comandi primitivi, ma a loro modo molto avanzati; all'occorrenza può tuttavia consultare il TEXbook.

Il primo argomento dell'ambiente *list* (attivato senza ricorrere a `\begin`, ma dando direttamente il comando 'interno' `\list`) non è racchiuso fra parentesi graffe, ma è il frutto di `\expandafter` e nuovamente di `\csname` e `\endcsname` che assieme agglutinano il prefisso `label` con il valore del contatore di annidamento espresso in numeri romani minuscoli.

Il secondo argomento dice di usare il contatore `\@enumctr` e dice di comporre l'etichetta di ogni voce mediante un numero composto in bandiera giustificata a destra, autorizzando anche di fuoriuscire dal margine di sinistra mediante l'accorgimento di far precedere il comando `\llap` (*left overlap*, sovrapponi a sinistra) con un blocco di gomma elastica di lunghezza naturale nulla e allungamento e accorciamento infiniti `\hss`.

Il comando di chiusura dell'ambiente *enumerate* è reso equivalente attraverso `\let` al comando di chiusura della generica lista `\endlist`.

L'esempio appena fatto, forse, è complicato, ma in realtà delucida tante cose che non sono state descritte nei dettagli negli appositi capitoli e mostra anche alcune tecniche di programmazione a livello di comandi primitivi di TeX.

Un esempio più semplice. Si vuole comporre una ambiente simile a *description* che abbia la possibilità di definire il margine sinistro sulla base della lunghezza di una parola; potrebbe essere utile per un glossario, per esempio. Vogliamo anche che ogni chiave che introduce ogni voce sia scritta in carattere lineare, ma non sia nero. Vediamo prima come è definito l'ambiente *description*:

```
\newenvironment{description}
               {\list{}{\labelwidth\z@ \itemindent-\leftmargin
                        \let\makelabel\descriptionlabel}}
               {\endlist}
\newcommand*\descriptionlabel[1]{\hspace\labelsep
                                 \normalfont\bfseries #1}
```

Dobbiamo quindi definire comandi simili, per esempio l'ambiente *descrizione*, che accetti un argomento sul quale prendere le misure del margine sinistro, e la sua etichetta `\etichettadescrizione`. Ecco come fare:

```
\newenvironment{descrizione}[1]%
      {\list{}{\settowidth{\labelwidth}{\normalfont\sffamily#1}%
                  \let\makelabel\etichettadescrizione}}
              {\endlist}
\newcommand*{\etichettadescrizione}[1]{\hspace\labelsep
                  \normalfont\sffamily #1}
```

Come si vede il compito era inizialmente più facile e la soluzione è decisamente più semplice.

29.5.3 Testo composto verbatim

Si possono usare due comandi e due ambienti per comporre del testo in modo verbatim; questo modo di comporre è quello che serve quando bisogna esporre,

per esempio, dei brani scritti in un linguaggio di programmazione nel quale si fa uso dei caratteri speciali di TEX (ovviamente con altri significati). I comandi sono

```
\verbsimbolo⟨testo da riprodurre⟩simbolo
oppure
\verb*simbolo⟨testo da riprodurre⟩simbolo
```

Il *simbolo* è un solo carattere e funziona da delimitatore del ⟨*testo da riprodurre*⟩; la versione senza asterisco riproduce gli spazi come spazi; la versione con l'asterisco riproduce gli spazi con il segno speciale ␣. Il ⟨*testo da riprodurre*⟩ deve comparire in una sola riga nel file sorgente.

Gli ambienti sono invece:

```
\begin{verbatim}⟨testo da riprodurre verbatim⟩\end{verbatim}
\begin{verbatim*}⟨testo da riprodurre verbatim⟩\end{verbatim*}
```

Il ⟨*testo da riprodurre verbatim*⟩ si può svolgere su diverse righe, anzi di solito è composto di diverse righe. L'unica riga che non si può riprodurre è `\end{verbatim}` con o senza asterisco. L'ambiente asteriscato, come il comando asteriscato, riproduce lo spazio in modo visibile con il carattere ␣.

Il comando `\verb` è fragilissimo e non può apparire come argomento di nessun altro comando (nemmeno nella versione asteriscata).

Il pacchetto standard *alltt* definisce un ambiente *alltt* da usare come l'ambiente *verbatim* dove tutti i caratteri sono diventati caratteri 'normali', salvo `\`, `{` e `}`.

Il pacchetto *fancyvrb* rende robusti il comando e l'ambiente con o senza asterisco, e aggiunge non poche funzionalità per rendere molto più utile l'uso di questi ambienti particolari; fra le altre cose permette di numerare le righe di una composizione di codice scritto in modo verbatim, permette di scrivere file in modo verbatim e di leggere file di codice senza eseguirne le macro che esso contiene.

29.6 Formule matematiche

29.6.1 Formule

La matematica in linea può essere delimitata dai seguenti delimitatori:

```
$ ⟨formula in linea⟩ $
\( ⟨formula in linea⟩ \)
\begin{math} ⟨formula in linea⟩ \end{math}
```

Si consiglia di non usare il primo metodo, anche se è più comodo da scrivere, perché si perde la diagnostica di LATEX nel caso che ci si dimentichi di uno dei

due delimitatori. L'ambiente *math*, come si può ben capire, non viene usato molto spesso. I delimitatori `\(` e `\)` sono i migliori, ma possono essere fragili, quando si compone con una vecchia versione di TeX Live. Invece i delimitatori `$...$` sono robusti.

Il comando

`\ensuremath{`⟨*formula*⟩`}`

permette di comporre una ⟨*formula*⟩ in linea garantendo che essa venga composta correttamente anche se il comando viene emesso in modo testuale. È comodo per definire macro che debbono potersi usare sia nel modo testo sia nel modo matematico.

Usando i pacchetti *babel* o *polyglossia* è disponibile il comando:

`\textormath`⟨*per il modo testo*⟩⟨*per il modo matematico*⟩

che produce un effetto analogo a quello di `\ensuremath`, con una notevole differenza; usato per definire un comando da usare sia in modo testo sia in modo matematico, può eseguire composizioni diverse nell'uno o nell'altro modo senza che il compositore debba preoccuparsi di verificare il modo di composizione; `\ensuremath`, invece, compone il suo argomento sempre in modo matematico, all'occorrenza entrando ed uscendo da questo modo, al fine di lasciare TeX nello stesso stato nel quale si trovava prima dell'esecuzione del comando.

Per la matematica in display si hanno gli ambienti

`\[` ⟨*formula in display*⟩ `\]`
`\begin{displaymath}` ⟨*formula in display*⟩ `\end{displaymath}`
`\begin{equation}` ⟨*formula in display*⟩ `\end{equation}`

I primi due ambienti non assegnano un numero alla ⟨*formula in display*⟩, mentre il terzo ambiente assegna un numero a detta formula. L'ambiente *displaymath* viene usato raramente visto che quello definito con `\[...\]` è molto più comodo da usare. L'ambiente *equation* non ha sostituti.

LaTeX offrirebbe anche gli ambienti *eqnarray* e *eqnarray** ma, per i motivi già esposti nel capitolo 14, è preferibile non servirsene; perciò, per comporre formule e sistemi di formule in display, è molto meglio servirsi del pacchetto *amsmath*.

Dentro gli ambienti *equation* e *eqnarray* (all'interno di ciascuna riga) è possibile inserire il comando `\label` così da definire una etichetta mnemonica per richiamare il numero della formula con `\ref` e/o con `\pageref`. Dentro *eqnarray* si può usare `\nonumber` per evitare che LaTeX assegni un numero ad una espressione che non si vuole numerare. Questo ambiente consentirebbe anche di spezzare una singola lunga espressione matematica su più righe, ricorrendo anche al comando suddetto e a `\lefteqn`; ma per questo scopo è meglio servirsi degli opportuni ambienti predisposti dal pacchetto *amsmath*.

Le espressioni in display usano un certo numero di parametri per comporre secondo diversi stili.

`\jot` serve per aggiungere altro spazio fra una equazione e l'altra in un sistema di equazioni.

`\mathindent` serve per fissare la rientranza sinistra delle equazioni quando queste sono composte allineate a sinistra se si usa l'opzione *fleqn* per la classe del documento.

`\abovedisplayskip` serve per definire il contrografismo che precede una equazione in display; viene usato un contrografismo più piccolo quando la riga prima del display è molto corta e l'espressione matematica potrebbe dare l'impressione ottica di essere preceduta da uno spazio troppo grande.

`\abovedisplayshortskip` è appunto questo contrografismo superiore più piccolo.

`\belowdisplayskip` è il contrografismo da inserire dopo una formula per distanziarla dal testo seguente.

`\belowdisplayshortskip` serve per inserire un contrografismo più piccolo se l'espressione matematica è seguita da una riga di testo molto corta.
Va da sé che i `...shortskip` non vengono usati quando si compone con l'opzione *fleqn* perché le espressioni sono tutte allineate a sinistra, a meno del loro rientro non molto importante (una cinquantina di punti, circa 15 mm).

Per comporre le espressioni matematiche si usano spesso piccoli comandi o piccole strutture di cui è facile dimenticarsi.

Esponenti gli esponenti si inseriscono mediante il comando `^` con la seguente sintassi:

```
^{⟨esponente⟩}
```

Con l'opzione *italian* di `babel` è disponibile il comando `\ap{⟨apice⟩}` per inserire un apice in tondo quando si è in modo matematico, oppure un apice con il font corrente quando si è in modo testo. Questo comando non è disponibile quando si usa `polyglossia`. Il nucleo di LaTeX per il modo testo prevede anche i comandi `\textsuperscript` `\textsubscript`, validi sempre, anche senza l'uso di `babel`. Solo pdflatex

Apici L'apice ′ o il doppio apice ″ si inseriscono con uno o due apostrofi, che in modo matematico vengono composti come apici.

Pedici e deponenti I pedici vengono inseriti con il comando `_`

```
_{⟨pedice⟩}
```

Con l'opzione *italian* di `babel` è possibile usare il comando `\ped{⟨pedice⟩}` per inserire un pedice in tondo in modo matematico e per inserire un deponente con il font corrente in modo testo. Anche questo comando non è disponibile quando si usa `polyglossia`. Solo pdflatex

Frazioni Le frazioni vengono composte con il comando

```
\frac{⟨numeratore⟩}{⟨denominatore⟩}
```

Bisogna ricordare che questo comando compone le frazioni in display come ci si aspetterebbe, ma le compone in modo testo sia nelle espressioni matematiche in linea, sia nelle sottoespressioni di espressioni in display, come per esempio in un altro numeratore o denominatore o come elemento di una matrice; in questi casi è preferibile servirsi delle frazioni a barra obliqua, piuttosto che di quelle a barra orizzontale.

Radici Le radici quadrate o con altro ⟨*indice*⟩ si compongono con

```
\sqrt[⟨indice⟩]{⟨radicando⟩}
```

Se l'⟨*indice*⟩ vale 2 non lo si indica.

Ellissi Le parti omesse (ellissi) vengono sostituite con i soliti tre puntini; in matematica se ne hanno di diversi tipi: `\dots` (…) si può usare sia in modo matematico sia in modo testo e produce i soliti puntini a livello della linea di base in modo testo, o a livello della linea di base o dell'asse matematico in modo matematico a seconda del contesto, cioè se è preceduto o seguito da segni di interpunzione appoggiati alla linea di base oppure di operatori binari centrati sull'asse matematico; `\ldots` produce lo stesso effetto di collocare i tre puntini sulla linea di base, ma è da usare solo in modo matematico; `\cdots` (⋯) produce i tre puntini allineati con il segno −, cioè sull'asse matematico della formula; `\vdots` (⋮) produce tre puntini verticali che servono per sostituire in verticale gli elementi omessi da colonne di matrici; infine `\ddots` (⋱) produce tre puntini in diagonale, utili per riempire una parte omessa dal cuore di una matrice.
Se si usa `\dots` in modo testo i tre puntini sono sempre sulla linea di base; ma se si usa il pacchetto *amsmath*, questo comando è ridefinito in modo che in matematica esso si comporti in modo intelligente, vale a dire esso colloca i puntini sulla linea di base se almeno da un lato è contornato da un segno di interpunzione, mentre esso produce i puntini sull'asse matematico della formula se è contornato da almeno un operatore binario. Si confrontino i seguenti due casi, composti entrambi con `\dots`:

$$i = 1, 2, 3, \dots, n \qquad i = 1 + 2 + 3 + \cdots + n$$

29.6.2 Simboli, accenti, delimitatori e grandi operatori

Tutti i simboli di qualunque genere usabili in matematica sono raccolti nelle tabelle 13.2–13.8; nelle tabelle 14.1–14.2, invece, sono raccolti gli ulteriori simboli disponibili con il pacchetto *amssymb*.

29.6.3 Impilare gli oggetti matematici

Una (breve) espressione può venire soprallineata con

```
\overline{⟨espressione⟩}
```

Analogamente si può sottolineare una espressione con

```
\underline{⟨espressione⟩}
```

Si possono marcare sopra o sotto alcune espressioni con delle graffe orizzontali che permettono di aggiungere altre informazioni all'espressione; si tratta dei comandi definiti dal pacchetto *amsmath* `\overset`, `\underset` e `\overunderset`; si veda la documentazione della versione aggiornata di *amsmath*; si veda anche il capitolo 30.

Anche gli accenti matematici possono venire impilati, ma si tratta semplicemente di racchiudere fra le graffe che delimitano l'argomento del primo accento una espressione già accentata.

Invece `\stackrel` permette di costruire degli operatori di relazione impilando uno sopra l'altro due segni distinti, scelti fra i vari simboli disponibili:

```
\stackrel{⟨elemento superiore⟩}{⟨elemento inferiore⟩}
```

29.6.4 Spaziatura matematica

Non si può raccomandare mai abbastanza il consiglio di non spaziare le espressioni matematiche. I comandi di spaziatura hanno senso solo se la spaziatura di default non è adeguata ai simboli che compaiono uno accanto all'altro in una specifica espressione matematica. Se, quindi, si inserirà una qualche spaziatura, lo si farà solo dopo aver corretto le bozze controllando accuratamente che questa spaziatura sia praticamente impercettibile.

`\,`	spazio sottile	`\:`	spazio medio
`\!`	spazio sottile negativo	`\;`	spazio grande

Il comando `\,` può essere usato anche in modo testo. Naturalmente si potrebbero anche usare `\quad` e `\qquad` oltre ai comandi di spaziatura del modo testo. Tuttavia merita segnalare i comandi primitivi `\mskip` e `\mkern` che accettano come argomenti (non racchiusi fra nessun tipo di parentesi) degli spazi espressi mediante le unità di misura chiamate `mu`, valide solo in matematica; 18 mu equivalgono a 1 em, ma in matematica l'unità 'em' cambia valore a seconda del font in uso, in particolare per il corpo principale della formula, per gli apici ed i pedici di primo ordine e per quelli di secondo ordine.

29.6.5 Font matematici

Per avere una intera formula composta con caratteri medi o con caratteri neri bisogna specificarlo prima di entrare in modo matematico:

```
\boldmath
⟨ambiente matematico⟩
\unboldmath
```

Altrimenti i vari simboli letterali e/o gli operatori possono essere resi con font diversi se si usano i comandi seguenti:

`\mathrm`	tondo	`\mathsf`	senza grazie
`\mathit`	*corsivo*	`\mathtt`	`spaziatura fissa`
`\mathbf`	**nero**	`\mathcal`	$\mathcal{CALLIGRAFICO}$

Per lualatex e xelatex

Usando i font OpenType con lualatex e xelatex si possono usare anche altri "alfabeti" (tutti racchiusi un uno unico file di segni matematici) con comandi simili ai precedenti; se ne può trovare l'elenco completo nella documentazione del pacchetto *unicode-math*, che d'altra parte va studiata con attenzione per poter usare le ulteriori possibilità offerte dai font OpenType.

Si noti che il corsivo matematico, il corsivo testuale usato in matematica mediante il comando `\mathit`, e il corsivo per il testo sono font con proprietà completamente diverse; per esempio il primo non contiene legature, per cui la parola *affine* verrebbe composta $af fine$. Più dettagliatamente gli esempi seguenti mostrano il diverso comportamento dei tre corsivi.

`\textit{affine, affine affine}`	*affine, affine affine*
`\( \mathit{affine, affine affine} \)`	$affine, affineaffine$
`\( affine, affine affine \)`	$af fine, af fineaf fine$

Si noti che i comandi testuali, come `\textit`, possono essere usati anche in matematica producendo semplice testo; mentre i comandi che scelgono i font in matematica lo fanno solo per le lettere e gli altri segni equivalenti alle lettere. Il corsivo matematico trae i suoi segni dallo stesso font usato per il testo corsivo ma con un file metrico completamente diverso. Si noti ancora che lo spazio dopo la virgola, nella prima riga dove si usa `\textit` appare esplicitamente come spazio interparola, mentre quando si scrive usando il comando `\mathit` nella seconda riga, o il corsivo matematico nella terza riga, si nota che la virgola è seguita dallo spazio sottile che appare in matematica a destra dei segni di interpunzione. Nella seconda riga, dove non c'è nessun segno di interpunzione fra la seconda e la terza istanza della parola *affine*, lo spazio, benché presente nel file sorgente, non è per nulla considerato, proprio come non è mai considerato in matematica. Nella seconda riga le legature sono rispettate con il corsivo testuale usato come font matematico, ma non appaiono nella terza riga con il corsivo matematico.

29.6.6 Stili di composizione matematica

I quattro stili di composizione della matematica sono

```
\displaystyle
\textstyle
\scriptstyle
\scriptscriptstyle
```

Lo stile `\textstyle` differisce dal `\displaystyle` nel senso che è più raccolto in verticale: gli apici e i pedici sono più vicini all'asse matematico; i limiti superiori e inferiori sono composti accanto all'operatore come se fossero apici o pedici; le frazioni sono composte in `\scriptstyle` per mantenere limitato l'ingombro verticale in modo che non sia necessario spaziare le righe del testo. È per questo che bisogna stare attenti, specialmente con le frazioni, a non renderle troppo piccole; piuttosto che una frazione troppo piccola, è preferibile una barra obliqua, senza rimpicciolire numeratori e denominatori, nemmeno quando sono puramente numerici, come in $^2/_3$; questa pratica, infatti, è 'deprecata' dalla norma UNI 2950.

29.7 Definizioni, numeri e programmazione

29.7.1 Comandi di definizione

Nuove macro possono essere definite, ridefinite o rese disponibili, se non lo sono già, mediante i comandi:

```
\newcommand{⟨comando⟩}[⟨numero argomenti⟩][⟨default⟩]{⟨definizione⟩}
\renewcommand{⟨comando⟩}[⟨numero argomenti⟩][⟨default⟩]{⟨definizione⟩}
\providecommand{⟨comando⟩}[⟨numero argomenti⟩][⟨default⟩]{⟨definizione⟩}
```

Sono tutti comandi fragili, ma del resto non sembra opportuno usarli negli argomenti di altri comandi. Sono già stati descritti in dettaglio nel capitolo 19. Tutti dispongono della versione asteriscata; la differenza sembra minima, ma è importante per evitare perdite di tempo quando si commettono errori nell'uso dei comandi definiti mediante queste istruzioni. In termini di gergo TEX, i comandi senza asterisco sono `\long`, mentre quelli con asterisco non lo sono. Un 'long command' che accetta uno o più argomenti è in grado di elaborare anche argomenti composti di più capoversi; per esempio `\parbox` è un 'long command'. I comandi che non hanno questo attributo non consentono che i loro argomenti contengano una linea vuota e/o un comando esplicito di fine capoverso, `\par`; se questo succedesse essi arresterebbero subito la compilazione con un messaggio di errore. Questo fatto succede abbastanza frequentemente quando ci si dimentica la parentesi graffa di chiusura di un argomento. Ma con i comandi 'corti' l'errore viene trovato subito con la normale diagnostica di LATEX, mentre con i comandi lunghi l'errore potrebbe venire segnalato o alla fine del file oppure perché la memoria degli *input buffer* del programma di composizione è satura.

Conviene esaminare la documentazione dei nuovi comandi basati sul linguaggio LATEX 3: `texdoc xparse`.

29.7.2 Comandi per la definizione di ambienti

Gli ambienti sono definiti o ridefiniti mediante i comandi

```
\newenvironment{⟨nome⟩}[⟨numero argomenti⟩][⟨default⟩]%
{⟨comandi di apertura⟩}{⟨comandi di chiusura⟩}
\renewenvironment{⟨nome⟩}[⟨numero argomenti⟩][⟨default⟩]%
{⟨comandi di apertura⟩}{⟨comandi di chiusura⟩}
```

Si veda il capitolo 19 per maggiori dettagli ed esempi.

29.7.3 Teoremi

LaTeX mette a disposizione dei comandi per definire degli pseudo ambienti al fine di comporre gli enunciati quali, per esempio, i teoremi. Il comando `\newtheorem` consente due forme di definizione:

```
\newtheorem{⟨nome⟩}{⟨titolino⟩}[⟨contatore dominante⟩]
\newtheorem{⟨nome⟩}[⟨numerato come⟩]{⟨titolino⟩}
```

dove:

⟨*nome*⟩ rappresenta il nome dello pseudoambiente; potrà essere, per esempio, teorema, lemma, definizione, corollario e simili. Per cui, se ⟨*nome*⟩ si riferisce a un `teorema`, il suo enunciato verrà racchiuso all'interno di `\begin{teorema}` ⟨*enunciato*⟩ `\end{teorema}`.

⟨*titolino*⟩ la parola che identifica l'enunciato; potrà essere Teorema, Lemma, Definizione,.... Non si confonda il nome dell'ambiente con il titolino dell'enunciato.

⟨*contatore dominante*⟩ è il contatore del capitolo, se si vuole che la numerazione del nuovo enunciato ricominci da 1 ad ogni nuovo capitolo; sarà il contatore dei paragrafi, se si desidera che la numerazione ricominci da 1 ad ogni nuovo paragrafo; eccetera.

⟨*numerato come*⟩ è invece il nome dello pseudoambiente del quale si vuole condividere la numerazione; questo deve essere già stato definito. Si potrebbe per esempio numerare con una sola sequenza numerica sia i teoremi sia i lemmi sia i corollari, mentre, probabilmente, si preferirebbe numerare separatamente le definizioni, le congetture, e altri simili enunciati.

Se non si specificano i contatori facoltativi, questi comandi definiscono un nuovo contatore con lo stesso nome dello pseudoambiente che può venire stampato usando il comando `\the`⟨*nome*⟩. Nella stessa maniera i comandi `\label` e `\ref` usano il nuovo nome del contatore per rendere accessibile ai riferimenti incrociati anche questi enunciati. Se invece si specifica il contatore ⟨*numerato come*⟩ non viene introdotto nessun nuovo contatore ma si usa quello già esistente. Infine se si specifica il nome del ⟨*contatore dominante*⟩, il comando `\the`⟨*nome*⟩ produrrà in stampa il numero del contatore relativo allo pseudoambiente preceduto dal numero stampato del ⟨*contatore dominante*⟩; se per esempio il ⟨*contatore dominante*⟩ fosse il paragrafo, allora il tredicesimo enunciato del quarto paragrafo del decimo capitolo verrà stampato nella forma 10.4.13.

29.8 Numeri, lunghezze e spazi

Il sistema TEX per comporre svolge una quantità di calcoli sia usando numeri veri e propri, sia usando lunghezze e spazi.

TEX fa calcoli solo con numeri interi; gli unici numeri non interi che esso gestisce sono i fattori moltiplicativi delle lunghezze o degli spazi. Qui di seguito vedremo come esistano registri per conservare i numeri, come agire sui numeri e sui loro registri, chiamati *contatori*.

TEX usa le lunghezze e dispone di registri per la conservazione di informazioni metriche; questi registri non hanno nomi particolari, come i contatori per i numeri, ma sono definiti da apposite sequenze di controllo con lo stesso nome; ma possono essere 'chiamati per nome'.

Gli spazi sono speciali lunghezze e i loro registri sono anch'essi un po' speciali; essi sono chiamati anche "lunghezze elastiche", perché hanno la proprietà di potersi allungare o accorciare rispetto alla loro lunghezza naturale; essi sono formati dalla collezione di tre lunghezze, la prima rappresenta la lunghezza naturale, la seconda l'ammontare assoluto di cui essa può allungarsi, e la terza l'ammontare assoluto con cui può restringersi.

Per operare con i numeri e le lunghezze rigide o elastiche TEX dispone di un certo numero di comandi primitivi. L'estensione `etex`, incorporata nelle versioni più recenti dei programmi di composizione, consente di eseguire certe operazioni in modo molto più comodo. Altrimenti, non disponendo di una versione recente di `pdftex`, bisogna ricorrere al pacchetto *`calc`*, alla cui documentazione si rimanda il lettore.

29.8.1 Numeri

A parte il modo di scrivere i numeri già visto nella pagina 725, LATEX consente di fare semplici conti con numeri interi e/o con le misure delle lunghezze.

`\newcounter` con la sintassi

> `\newcounter{`⟨*nome*⟩`}[`⟨*contatore dominante*⟩`]`

permette di definire un nuovo contatore per numeri interi chiamato ⟨*nome*⟩, asservito al ⟨*contatore dominante*⟩; quindi viene azzerato, quando il ⟨*contatore dominante*⟩ viene incrementato. Il ⟨*nome*⟩ e il ⟨*contatore dominante*⟩ hanno lo stesso significato descritto per i teoremi.

`\setcounter` con la sintassi

> `\setcounter{`⟨*nome*⟩`}{`⟨*valore*⟩`}`

inserisce la quantità numerica ⟨*valore*⟩ nel contatore che ha quel ⟨*nome*⟩.

`\addtocounter` con la sintassi

> `\addtocounter{`⟨*nome*⟩`}{`⟨*valore*⟩`}`

aggiunge il ⟨*valore*⟩ specificato al contenuto del contatore ⟨*nome*⟩. Il ⟨*valore*⟩ può anche essere negativo, quindi di fatto LaTeX esegue una somma algebrica.

`\value` con la sintassi

```
\value{⟨nome⟩}
```

recupera il contenuto del contatore ⟨*nome*⟩ per passarlo ad altri comandi o per rendere stampabile il numero contenuto dentro quel contatore.

`\stepcounter` con la sintassi

```
\stepcounter{⟨nome⟩}
```

incrementa il contatore ⟨*nome*⟩ di una unità.

`\refstepcounter` con la sintassi

```
\refstepcounter{⟨nome⟩}
```

oltre ad incrementare il contatore ⟨*nome*⟩ di una unità lo rende accessibile al comando `\label` così che si possa fare riferimento ai valori del contatore mediante le chiavi usate da `\label` e `\ref`.

`\arabic` con la sintassi

```
\arabic{⟨nome⟩}
```

trasforma il contenuto del contatore ⟨*nome*⟩ nell'equivalente stringa di cifre arabe.

`\roman` con la sintassi

```
\roman{⟨nome⟩}
```

trasforma il contenuto del contatore ⟨*nome*⟩ nell'equivalente stringa di cifre romane minuscole; nel testo si è già commentato a proposito dell'infelice scelta delle cifre tonde minuscole, mentre sarebbe meglio ricorrere sempre alle lettere minuscole del maiuscoletto; questo però non è disponibile in tutte le famiglie, serie e forme, quindi l'utente deve provvedere a mano caso per caso.

`\Roman` con la sintassi

```
\Roman{⟨nome⟩}
```

trasforma il contenuto del contatore ⟨*nome*⟩ nell'equivalente stringa di cifre romane maiuscole.

`\alph` con la sintassi

```
\alph{⟨nome⟩}
```

trasforma il contenuto del contatore ⟨*nome*⟩ nella lettera dell'alfabeto latino minuscolo corrispondente alla posizione indicata dal contatore; ovviamente in questo modo si può rappresentare il valore del contatore solo finché il suo contenuto è positivo ma non superiore a 26, essendo 26 le lettere dell'alfabeto latino.

`\Alph` con la sintassi

```
\Alph{⟨nome⟩}
```

trasforma il contenuto del contatore ⟨*nome*⟩ nella lettera dell'alfabeto latino maiuscolo corrispondente alla posizione indicata dal contatore; ovviamente in questo modo si può rappresentare il valore del contatore solo finché il suo contenuto è positivo ma non superiore a 26, essendo 26 le lettere dell'alfabeto latino.

`\fnsymbol` LaTeX contiene una lista ordinata di simboli, ∗, †, ‡, §, ¶, ‖, ∗∗, ††, ‡‡, dalla quale il comando `\fnsymbol`, con la sintassi

```
\fnsymbol{⟨nome⟩}
```

trae il simbolo corrispondente alla posizione indicata dal contatore ⟨*nome*⟩; questo comando serve solo per i richiami di nota del comando `\thanks`, che può essere usato solo per comporre la lista degli autori, il titolo e la data quando si usa il comando `\maketitle` o quando si usa l'ambiente *titlepage*. Indicando le note con questi simboli si possono rendere univoche le note con un numero positivo, non superiore a 9, visto che la lista contiene solo nove simboli.

Quando si definisce un nuovo contatore ⟨*nome*⟩ LaTeX definisce anche il modo di stamparlo attraverso il comando `\the`⟨*nome*⟩; in altre parole se si definisce il nuovo contatore `pippo` mediante il comando:

```
\newcounter{pippo}
```

viene contemporaneamente definito il comando `\thepippo`, in generale in modo molto semplice[1]:

```
\newcommand\thepippo{\arabic{pippo}}
```

Il compositore è libero di ridefinire il modo di scrivere il contatore, specialmente se è asservito ad un altro contatore; a titolo di esempio scriverà:

```
\newcounter{pippo}[pluto]
\renewcommand*\thepippo{\arabic{pluto}.\arabic{pippo}}
```

[1] In realtà la definizione normalmente viene eseguita ricorrendo ai comandi primitivi che sono più veloci da eseguire in quanto non hanno bisogno di essere interpretati.

se vuole che il contatore `pippo`, supponendo che valga 3, quando il contatore `pluto` vale 5 venga stampato nella forma 5.3.

Vale la pena di segnalare che il comando primitivo `\the` può precedere il nome di qualunque registro interno dell'interprete per trasformarne il contenuto in una stringa di caratteri che ne indica il contenuto. Questo comando primitivo non è applicabile ai nomi dei contatori LaTeX, perché i loro nomi non individuano direttamente il registro, come invece succede con le lunghezze. Quando si definisce un contatore LaTeX, viene associato un particolare registro numerico, il primo disponibile, ad una sequenza formata con il ⟨*nome*⟩ preceduto alla stringa `\c@`. Quindi il nome del contatore LaTeX `pippo` serve per definire la sequenza `\c@pippo` che viene associata all'indirizzo numerico del registro interno (di cui il compositore non deve preoccuparsi). Il comando `\the` dovrebbe quindi precedere la sequenza `\c@pippo` così:

```
\the\c@pippo
```

per produrne la stringa di cifre che ne indica il contenuto. Data la presenza del segno `@`, che normalmente non è una lettera, anche se può essere usato come lettera all'interno dei file di classe o di quelli di estensione, questo modo di procedere è fortemente sconsigliato; tra l'altro è molto più comodo, e meno incline agli errori, ricorrere ai comandi specifici, come per esempio `\arabic`, per scrivere il contenuto dei contatori LaTeX; in ogni caso il valore numerico contenuto dentro un registro, indipendentemente dal modo di scriverlo può sempre essere usato tramite l'intermediario del comando `\value`: `\value{pippo}` restituisce il valore numerico contenuto dentro il contatore LaTeX `pippo`, e questo valore può venire usato come meglio si crede, anche come argomento dei test numerici `\ifnum`, `\ifodd` e simili.

29.8.2 Lunghezze

Le lunghezze esplicite, quelle, cioè, che si esprimono con una misura e una unità di misura, sono costituite da una sequenza di cifre decimali, separate dal separatore decimale che in teoria potrebbe anche essere la virgola; siccome nella sintassi LaTeX la virgola svolge anche altri ruoli, è preferibile usare sempre e soltanto il punto decimale quando si esprimono le misure delle lunghezze.

Un numero che contenga una parte intera nulla potrebbe cominciare con il punto decimale, omettendo lo zero che dovrebbe precederlo; si sconsiglia questa abbreviazione del tutto irrilevante ma che facilita gli errori. Il numero può essere preceduto dal segno positivo o negativo; quello positivo è evidentemente facoltativo; quello negativo invece non lo è. Se in seguito alla sostituzione di espressioni ed argomenti nelle macro la misura risultasse preceduta da diversi segni, il numero di segni negativi si comporta come ci si aspetterebbe, nel senso che ognuno cambia segno a quanto segue, quindi un numero dispari di segni negativi porta ad un risultato complessivamente negativo, mentre un numero pari porta ad un risultato positivo.

Le lunghezze definite con i comandi LaTeX sono sempre lunghezze elastiche, anche se i loro allungamenti e accorciamenti sono nulli; di fatto si comportano come lunghezze rigide, a meno che nella definizione non vengano specificati anche l'allungamento e/o l'accorciamento.

Unità di misura Le unità di misura accettabili da LaTeX sono le seguenti:

- `cm` centimetri.
- `mm` millimetri.
- `in` pollici.
- `pt` punti tipografici (1 pt = (1/72,27) in = 0,3515 mm).
- `pc` pica (1 pc = 12 pt).
- `bp` punto PostScript o *big point* (1 bp = (1/72) in = 0,3528 mm).
- `sp` *scaled point* (la frazione 2^{-16} di un punto tipografico).
- `dd` punto didot (1 dd=0,3760 mm=1,070 pt).
- `cc` cicero (1 cc=12 dd).
- `ex` *x-height*, occhio del carattere corrente, altezza della lettera 'x'.
- `em` *em-width*, larghezza della 'M', convenzionalmente circa uguale al corpo del font corrente; questa convenzione è approssimativamente valida per i font proporzionali, mentre non è valida per i font monospaziati.

`\fill` è una lunghezza elastica (gomma) di lunghezza naturale nulla ma infinitamente allungabile.

`\stretch{`⟨*moltiplicatore*⟩**`}`** è una lunghezza elastica (gomma) di lunghezza naturale nulla ma allungabile infinitamente quanto `\fill` moltiplicata per ⟨*moltiplicatore*⟩; questo è un numero decimale con segno facoltativo; se il segno è negativo questa lunghezza diventa 'infinitamente' accorciabile.

`\newlength{`⟨*comando*⟩**`}`** definisce una nuova lunghezza identificata con ⟨*comando*⟩; questo è il nome di un tipico comando LaTeX formato da una stringa letterale preceduta dal backslash, oppure da un solo segno non letterale preceduto dal backslash. Si possono definire fino a 256 lunghezze; in realtà i motori di composizione del sistema TeX (sufficientemente recenti) accettano la definizione di un numero molto maggiore di lunghezze, ma per per usare `\newlength` al fine di dare un nome a un registro lunghezza con un numero maggiore di 255, diventa necessario caricare il pacchetto ***`etex`***.

`\setlength{`⟨*comando*⟩**`}{`**⟨*lunghezza*⟩**`}`** serve per assegnare la ⟨*lunghezza*⟩ esplicita al registro-lunghezza identificato con ⟨*comando*⟩.

`\addtolength`⟨*comando*⟩⟨*lunghezza*⟩ serve per incrementare il valore di lunghezza contenuto nel registro ⟨*comando*⟩ della quantità ⟨*lunghezza*⟩, che può essere sia positiva, sia negativa; in questo caso si ha una sottrazione.

`\settowidth, \settoheight e \settodepth` seguono la sintassi seguente:

```
\settowidth{⟨comando⟩}{⟨testo⟩}
\settoheight{⟨comando⟩}{⟨testo⟩}
\settodepth{⟨comando⟩}{⟨testo⟩}
```

Ognuna di queste istruzioni assegna al registro-lunghezza ⟨*comando*⟩ rispettivamente la larghezza, l'altezza o la profondità della stringa che costituisce il ⟨*testo*⟩.

29.8.3 Operazioni fra numeri e grandezze: le estensioni di etex e il pacchetto *xfp*

Le recenti estensioni del programma di composizione del sistema TeX includono tutte le estensioni del precedente `etex`; una descrizione esaustiva è contenuta nel documento `.../textmf-distr/doc/etex/base/etex_man.pdf` (semplicemente leggibile dando il comando da terminale `texdoc etex`. Quello che ora ci interessa di più è la sezione 3.5 di quel documento intitolata "Expressions".

Vediamo dunque che ora ogni programma è in grado di calcolare alcune espressioni matematiche direttamente in memoria (nei registri della CPU del calcolatore che si sta usando). Il risultato di queste espressioni è usabile in qualunque momento sia lecito usare un risultato numerico o dimensionale, per esempio assegnandolo ad un registro o inserendolo in una espressione logica.

Le espressioni dimensionali `\dimexpr` e quelle numeriche `\numexpr` possono venire mescolate; non è sempre obbligatorio terminare ogni espressione di ciascun tipo con `\relax` ma è consigliabile. Esistono espressioni anche per le lunghezze elastiche e per le lunghezze matematiche: `\glueexpr` e `\muexpr`. Si possono usare le parentesi tonde per alterare la sequenza di esecuzione delle operazioni nello stesso modo in cui le parentesi vengono usate in matematica. L'esempio che il testo propone è il seguente:

```
\ifdim\dimexpr (2pt-5pt)*\numexpr 3-3*13/5\relax + 34pt/2 < \wd20
```

Vi compare una espressione mista dimensionale che contiene al suo interno una espressione numerica; il comando `\relax` serve per terminare l'espressione numerica; quella dimensionale termina con il primo carattere non valido in una espressione, vale a dire che termina subito prima del segno di minore `<`.

Le operazioni valide all'interno dell'espressione sono i segni delle quattro operazioni aritmetiche: `+`, `-`, `*`, e `/`

Le operazioni numeriche sono eseguite fra numeri interi (o con l'indicazione di contatori TeX oppure contatori LaTeX passati come argomento di `\value`). Le divisioni, quindi, sono divisioni intere, cioè forniscono solo la parte intera del quoziente che potremmo calcolare con una calcolatrice ordinaria (o anche a mano) ma per ottenere il risultato intero, specialmente delle divisioni, il risultato viene *arrotondato* all'intero più vicino, mentre `tex` calcola il quoziente intero *troncandolo*.

Le operazioni dimensionali sono le stesse e sono del tutto equivalenti a quelle che potrebbero venire eseguite usando i comandi primitivi di `pdftex`, vale a dire `\advance` per la somma algebrica, `\multiply` per la moltiplicazione e `\divide` per la divisione intera; anche in questo caso il risultato delle operazioni dimensionali viene arrotondato allo *scaled point* più vicino; sembra un'inezia, ma bisogna

tenerne conto per impostare i test dimensionali, perché questo arrotondamento potrebbe dare luogo a risultati contrari a quelli che ci spetteremmo; non è un inconveniente insolito; è comune a tutti i processi di calcolo che prevedono arrotondamenti, anche lavorando con programmi diversi da quelli del sistema TEX, come il C, il C++, il FORTRAN, ed altri programmi di calcolo.

C'è però una novità: le operazioni di scalamento di una lunghezza sono intese come il prodotto della lunghezza da scalare per il numeratore del fattore di scala con il risultato diviso per il denominatore del fattore di scala; numeratore e denominatore possono essere numeri interi (eventualmente contenuti in contatori) oppure altre due lunghezze; non possono essere uno un numero e l'altro una lunghezza, ma devono essere entrambi dello stesso tipo.

Bene questa operazione di scalamento viene fatta mettendo il risultato della prima moltiplicazione in un registro della CPU a 64 bit, cioè lungo il doppio di una normale parola di 4 byte, e il risultato della successiva divisione viene riportato ad una parola di 32 bit, cioè di 4 byte. Nel fare questo si perdono dei bit nella parte meno significativa della misura della lunghezza, e pdftex non esegue un semplice troncamento, ma esegue l'arrotondamento degli *scaled points* al valore intero più vicino e quindi il valore in punti tipografici alla quinta cifra decimale più vicina.

Vediamo un esempio: se la giustezza `\textwidth` dovesse venire divisa per sei al fine di comporre a sei colonne, bisogna togliere cinque volte lo spazio intercolonna e dividere il risultato per sei; il tutto va poi assegnato alla lunghezza `\columnwidth`; scriveremo allora[2]:

```
\newlength\Numer \newlength\Denom
...
\setlength\Numer{1pt} \setlength\Denom{6pt}
\columnwidth=\dimexpr(\textwidth -5\columnsep)*\Numer/\Denom\relax
```

Si noti che moltiplicare per 1 pt e dividere per 6 pt non è la stessa cosa di moltiplicare per 1 e dividere per 6. Nella codifica interna la lunghezza di 1 pt è data dal numero intero di *scaled points* corrispondenti appunto a 1 pt, quindi da un numero binario formato da un 1 seguito da sedici zeri binari; analogamente il numero sei è rappresento da tre cifre binarie `110`, seguite da sedici zeri binari; la prima moltiplicazione pertanto sposta a sinistra di sedici posizioni il numero del risultato della prima sottoespressione, poi viene diviso per sei e il risultato viene scalato a destra di sedici posizioni, portando ad un suo eventuale arrotondamento. Moltiplicare semplicemente per uno e dividere semplicemente per sei non porta con sé nessuno scalamento e nessun arrotondamento, ma solo un eventuale troncamento. Infatti, eseguendo l'operazione indicata sopra, l'espressione calcolata con `\dimexpr` vale 49,16667 pt, mentre quella calcolata con gli operatori primitivi vale 49,16666 pt. In questo caso la differenza è trascurabile, ma in altri casi, specialmente eseguendo dei test, potrebbe non esserlo.

[2] Usando la sintassi di basso livello dei programmi di composizione.

Si noti infine che le operazioni di scalamento vengono eseguite correttamente se si usano i registri lunghezza (o i registri numerici) piuttosto che le grandezze esplicite.

Vale la pena di indicare alcune particolarità dei calcoli che il programma esegue e dei registri in cui salva certe quantità; è importante anche al fine di rendersi conto della delicatezza dei calcoli e dell'eventualità di eseguirne alcuni che portano o alla perdita di ogni informazione o all'*overflow*.

I registri interi, cioè i contatori sono parole di 4 byte, che contengono 32 bit o cifre binarie. Un bit è riservato per il segno, quindi restano 31 bit per rappresentare i numeri. Ne segue che il massimo numero intero vale $2^{31} - 1 = 2\,147\,483\,647$; chiaramente un numero sufficientemente grande da dubitare di poterlo superare.

I registri per le lunghezze sono ugualmente dei registri interi dove viene memorizzato il numero di *scaled points* corrispondenti. Si tratta di parole di 4 byte che contengono 32 bit; due di questi bit servono per il segno e per altre informazioni specifiche relative alle lunghezze; restano quindi 30 bit per memorizzare la grandezza espressa mediante un numero intero di *scaled points* il cui valore assoluto massimo può quindi arrivare a 1 073 741 823. Siccome 2^{16} sp corrispondono a 1 pt, la massima lunghezza espressa in punti che può venire memorizzata vale 16 383,99999 pt. Questo valore, tra l'altro viene memorizzato nel registro di lunghezza `\maxdimen`, nel caso il compositore/programmatore volesse eseguire dei confronti o utilizzare questo numero speciale.

Questa massima lunghezza corrisponde a poco più di 5758 mm, quasi sei metri e sembra che anche questo valore in tipografia sia difficilmente raggiungibile. Purtroppo non è vero. Si pensi infatti ai calcoli che bisognerebbe fare per determinare una lunghezza che sia in proporzione con un'altra lunghezza con lo stesso rapporto di altre due lunghezze:

$$\frac{l_1}{l_2} = \frac{L_1}{L_2} \qquad \text{cioè} \qquad l_1 = l_2 \frac{L_1}{L_2}$$

Per fare un esempio pratico, si supponga di avere a disposizione una carta di formato non standard con base L_2 e altezza L_1 da specificare nel preambolo assegnandone i valori ai registri di lunghezza `\paperwidth` e `\paperheight`; in base al font usato si determina che la leggibilità è ottimale se la base della griglia interna `\textwidth` vale l_2; quanto deve valere `\textheight` affinché il testo sia inscritto in un rettangolo simile al rettangolo della carta?

Con le espressioni di dimensione che abbiamo appena visto possiamo determinare:

```
\textheight=\dimexpr \textwidth * \paperheight / \paperwidth \relax
```

e il programma ci esegue i calcoli senza problemi grazie ai risultati interni conservati un un registro di due parole, cioè di 64 bit.

Cosa sarebbe successo se avessimo usato i comandi primitivi `\multiply` e `\divide`? Per rendercene conto è meglio fare un esempio numerico. Supponiamo che la carta abbia le dimensioni di 220 mm per 280 mm (il vecchio formato italiano quadrotto, simile al formato letter americano, ma che in Italia non è più

facilmente reperibile, se non addirittura irreperibile). Il calcolo della giustezza sulla base dei font che andiamo ad usare ci dice che questa dovrebbe valere 170 mm. Eseguendo i calcoli con una calcolatrice tascabile (a virgola mobile) avremmo[3]:

$$x = l_2 \cdot L_1 = 170 \times 280 = 47600 \gg 5758$$
$$l_1 = x/L_2 = 47600/220 = 216{,}364$$

Come si vede il risultato finale è perfettamente accettabile, ma la moltiplicazione intermedia eccede le possibilità dei registri interni di pdftex e quindi si ha un *overflow*.

Se è così, allora eseguiamo prima la divisione e poi la moltiplicazione:

$$y = l_2/L_2 = 170/220 = 0$$
$$l_1 = y \cdot L_1 = 0 \times 280 = 0$$

ma in questo modo la divisione intera di un numero per un numero più grande ci dà la parte intera del quoziente che vale evidentemente zero e perdiamo ogni informazione.

Ecco, quindi, dove nasce l'enorme comodità dell'operazione di scalamento eseguita dal programma in un registro interno a 64 bit. Questo genere di cose possono però succedere anche con il programma quando si esegue la divisione finale, se il divisore è troppo piccolo. In questo caso il programma non può fare a meno di emettere un messaggio di errore, ma si può proseguire con le dita incrociate; i calcoli saranno sbagliati, quindi andrà corretto l'errore prima di ottenere risultati ragionevoli, ma lo si potrà fare senza perdere tutte le informazioni accessorie che il programma è in grado di raccogliere o di generare durante la sua esecuzione.

Inconvenienti di questo genere possono succedere più sovente di quanto si immagini, quindi è opportuno fare sempre una analisi preventiva dei calcoli da eseguire, almeno come ordini di grandezza, per non ritrovarsi senza la possibilità di lasciare fare i conti al programma al fine di creare macro capaci di funzionare con qualunque insieme di valori degli argomenti.

Merita ricordare che la recente (2018) introduzione del pacchetto *xfp*, le cui funzionalità sono ora integrate nel nucleo di LaTeX, permette di fare molto di più con la matematica decimale, invece che binaria, con numeri e dimensioni in virgola mobile (e anche in notazione scientifica), Non esegue solo le quattro operazioni, ma anche le radici e le funzioni della serie esponenziale/logaritmica, e le funzioni trigonometriche dirette e inverse sia in radianti sia in gradi; permette di arrotondare il risultato limitandone il numero di cifre decimali e producendo a

[3]Non si badi al fatto che le indicazioni contengano solo le misure e non le unità di misura. È voluto per due motivi: (*a*) l'elaboratore usa solamente numeri, e (*b*) qui non si vuole annoiare il lettore con lunghe scritture di sequenze di bit o di cifre esadecimali od ottali; il concetto è quello di mostrare come si svolgono i conti, anche se qui usiamo le misure dei millimetri, non dimentichiamoci che i contatori dimensionali non possono contenere un numero equivalente di millimetri superiore a poco più di 5758.

propria scelta sia gli arrotondamenti, sia i troncamenti, sia le operazioni indicate con i nomi inglesi *ceil* (troncamento verso lo zero) e *floor* (arrotondamento allontanandosi da zero) in modo da gestire correttamente sia i numeri positivi sia quelli negativi. Le funzionalità del pacchetto **xfp** funzionano correttamente solo con una versione della distribuzione del sistema TeX dal 2018 in poi e richiedono anche un file di formato LaTeX corrispondentemente aggiornato. Il pacchetto è ancora in via di sviluppo, ma quanto descritto è disponibile già dalla sua prima versione del 2018.[4]

29.8.4 Il pacchetti *ifthen* e *etoolbox*

Il pacchetti *ifthen* e *etoolbox* permettono di accedere ai comandi primitivi di controllo del flusso delle informazioni di cui TeX è capace. Il pacchetto *ifthen* fa parte di ogni distribuzione anche minimale del sistema TeX, mentre *etoolbox* non ne fa parte e, se la distribuzione non è completa o almeno quasi completa, bisogna installarlo apposta. Questo potrebbe essere un piccolo problema per gli utenti della distribuzione MiKTeX, che spesso dispongono di una distribuzione minimale o poco più. Alla fine del paragrafo si cercherà di illustrare un poco *etoolbox* che è una "scatola di attrezzi" molto completa e non gestisce solo i comandi condizionali, ma gestisce anche una miriade di altre funzionalità.

29.8.4.1 Il pacchetto *ifthen*

I comandi messi a disposizione da *ifthen* sono i seguenti.

`\ifthenelse` con la sintassi:

> `\ifthenelse{`⟨*test*⟩`}{`⟨*esegui quando è vero*⟩`}{`⟨*esegui quando è falso*⟩`}`

esegue il ⟨*test*⟩ e se questo ⟨*test*⟩ restituisce il valore 'vero' allora vengono passati al flusso di informazioni da elaborare i token che formano il contenuto del primo argomento dopo il test, ⟨*esegui quando è vero*⟩. Altrimenti vengono eseguiti i token che formano il testo di ⟨*esegui quando è falso*⟩.
I ⟨*test*⟩ che si possono eseguire sono i seguenti.

confronto numerico con la sintassi

> ⟨$numero_1$⟩ ⟨*operatore*⟩ ⟨$numero_2$⟩

dove ⟨*operatore*⟩ è uguale a `>`, oppure `=`, oppure `<`; i numeri ⟨$numero_i$⟩ possono essere i contenuti di due contatori o di un contatore e di una quantità esplicita. Per i contatori LaTeX bisogna specificarne il valore attraverso la macro `\value`.

[4] Chi scrive ha già avuto modo di usarlo ottenendo un notevole risparmio di codice e una maggiore precisione. Il pacchetto **xfp** implementa una funzione arcotangente a due argomenti, che permette di calcolare l'anomalia di un numero complesso nell'intervallo $-180° < \varphi \leq +180°$. Questa ottiene le anomalie giuste come esemplificato qui di seguito:

Numero complesso	$0+\mathrm{i}0$	$1-\mathrm{i}0$	$1+\mathrm{i}1$	$0+\mathrm{i}1$	$-1+\mathrm{i}1$	$-1,+\mathrm{i}0$	$-1,-\mathrm{i}1$	$0,-\mathrm{i}1$	$1,-\mathrm{i}1$
Anomalia in gradi	0	0	45	90	135	180	−135	−90	−45

`\equal` con la sintassi:

> `\equal{`⟨*stringa*$_1$⟩`}{`⟨*stringa*$_2$⟩`}`

confronta due stringhe e se sono assolutamente identiche il test è vero, altrimenti è falso.

`\lengthtest` con la sintassi:

> `\lengthtest{`⟨*lungh*$_1$ ⟨*operatore*⟩ *lungh*$_2$⟩`}`

confronta due lunghezze (generalmente almeno una delle due è contenuta in un registro-lunghezza) e restituisce il valore 'vero' se le due lunghezza stanno nella relazione implicata dall'⟨*operatore*⟩. Questo può essere un solo segno matematico fra `=`, `>`, oppure `<`.

`\isodd` con la sintassi:

> `\isodd{`⟨*numero*⟩`}`

controlla se un numero (generalmente contenuto in un contatore, eventualmente ottenuto tramite il comando `\value` per i contatori LaTeX) sia dispari.

`\isundefined` con la sintassi:

> `\isundefined{`⟨`\`*comando*⟩`}`

restituisce il valore vero se ⟨`\`*comando*⟩ non è mai stato definito.

`\boolean` controlla lo stato di una variabile booleana; la sintassi per gestire queste variabili è la seguente.

> `\newboolean{`⟨*nuova variabile booleana*⟩`}`
> `\provideboolean{`⟨*nuova variabile booleana*⟩`}`
> `\setboolean{`⟨*variabile booleana*⟩`}{`⟨*stato*⟩`}`
> `\boolean{`⟨*variabile booleana*⟩`}`

dove ⟨*stato*⟩ è una delle due parole `true` (vero) oppure `false` (falso). In realtà `\boolean` è in grado di verificare lo stato anche delle variabili booleane interne a LaTeX, e anche di quelle definite con i comandi elementari di TeX o di Plain TeX. Con quest'ultimo si definisce un nuovo comando logico con `\newif` il quale accetta come argomento il comando per disteso e contemporaneamente definisce due altri comandi per impostare le corrispondenti variabili booleane. In pratica, se si volesse definire una nuova variabile booleana '`test`', con il pacchetto *`ifthen`* si dovrebbero usare i comandi

```
\newboolean{test}
...
\setboolean{test}{true}
...
\ifthenelse{\boolean{test}}{\textbf{Pippo}}{\textit{Pluto}}
```

Se invece si usassero i comandi di Plain TeX (accessibili anche quando si usa LaTeX) si dovrebbe scrivere

```
\newif\iftest
...
\testtrue
...
\iftest\textbf{Pippo}\else\textit{Pluto}\fi
```

Si è fatto questo esempio non tanto per invitare ad usare i comandi elementari di Plain TeX, quanto per permettere di capire come funzionano i test elementari che si trovano scritti a piene mani nei comandi definiti nei file di formato, di classe e di estensione.
Il comando `\provideboolean` esegue la stessa definizione che esegue `\newboolean`, ma omettendo ogni definizione se la ⟨*variabile booleana*⟩ esiste già.

`\and` `\or` e `\not` permettono di mettere insieme diverse frasi logiche da collegare fra di loro mediante questi operatori; l'intera frase comprendente gli operatori deve essere racchiusa fra `\(` e `\)`.

`\whiledo` consente di descrivere e realizzare un ciclo 'while'; la sintassi è:

`\whiledo{`⟨*test*⟩`}{`⟨*ciclo*⟩`}`

Questo comando ripete il ⟨*ciclo*⟩ fino a quando il ⟨*test*⟩, inizialmente vero, diventa falso. Va da sé che, prima di iniziare la ripetizione di ⟨*ciclo*⟩, gli elementi da cui dipende ⟨*test*⟩ devono essere inizializzati in modo tale che ⟨*test*⟩ sia vero. Il ⟨*ciclo*⟩ deve contenere delle istruzioni o dei comandi che prima o poi rendano il ⟨*test*⟩ falso, altrimenti LaTeX entra in un ciclo infinito e non ne esce più.

A titolo di esempio dell'uso di `\ifthenelse`, il comando `\cleardoublepage` è definito nel nucleo di LaTeX mediante la sintassi di basso livello TeX:

```
\def\cleardoublepage{\clearpage\if@twoside \ifodd\c@page\else
    \hbox{}\newpage\if@twocolumn\hbox{}\newpage\fi\fi\fi}
```

Usando il pacchetto *ifthen* questa definizione potrebbe essere tradotta in:

```
\newcommand*\cleardoublepage%
{\clearpage
  \ifthenelse{\(\boolean{@twoside}\and\not\isodd{\value{page}}\)}%
             {%
                \mbox{}\newpage
                \ifthenelse{\boolean{@twocolumn}}{\mbox{}\newpage}{}%
             }{}%
}
```

La scrittura è un poco più complessa, ma si vedono meglio gli effetti e la natura dei test; il primo controlla se stiamo componendo fronte e retro e se siamo su una pagina pari; è chiaro che se non stiamo componendo fronte e retro, saltare una pagina per ricominciare da una pagina dispari, una pagina di destra, non ha molto senso; inoltre è chiaro che se la pagina è dispari non bisogna fare nulla. Infine controlla se si sta componendo su due colonne: in questo caso esegue comunque un `\newpage` che, a due colonne, vuol dire di interrompere una colonna e ricominciare a comporre nella colonna successiva[5]. L'incolonnamento dei contenuti delle varie clausole e delle relative parentesi graffe permette di seguire meglio i successivi passi.

29.8.4.2 Il pacchetto *etoolbox*

Il pacchetto *etoolbox* dispone di una miriade di nuovi comandi che certamente non verranno descritti qui di seguito, tranne alcuni che abbiano a che fare con i comandi condizionali. Queso pacchetto dispone di comandi per svolgere azioni importanti nei seguenti settori:

Definizioni di comandi estende le definizioni standard di LaTeX provvedendo anche a definizioni espanse e globali; provvede a definire comandi robusti e a "irrobustire" comandi fragili preesistenti.

Riparazioni o modifiche di codice provvede ad estendere o comunque a modificare il testo sostitutivo di macro esistenti.

Protezione di comandi Estende i meccanismi di protezione di LaTeX.

Gestione di lunghezze e contatori non solo estende la gestione delle lunghezze e dei contatori, ma permette di definire un numero maggiore di registri e di contatori ad un numero talmente alto che in pratica si può considerare illimitato. Toglie la barriera dei 256 contatori e registri di ogni tipo che potevano essere usati con i programmi del sistema TeX; questo limite non esiste più, come spiegato nel capitolo 30.

Agganci ai comandi di gestione del documento LaTeX dispone di pochi comandi, come per esempio `\AtBeginDocument`, che possano differire l'esecuzione di certi comandi a posizioni speciali della gestione dei file sorgente; *etoolbox* estende enormemente queste capacità.

Agganci agli ambienti in modo analogo vengono definiti altri "ganci" (in inglese *hooks*) che permettono di gestire correttamente le istruzioni da eseguire al momento opportuno dentro a un ambiente o "ai suoi estremi".

Comandi ad uso degli autori Mentre i comandi precedenti sono di interesse specialmente per chi scrive file di classe o di estensione, i prossimi comandi sono particolarmente utili per gli autori.

Definizioni vengono messi a disposizione ulteriori comandi per gestire le definizioni di comandi.

[5] Il secondo `\ifthenelse` può essere omesso se il comando precedente, invece di `\newpage`, fosse nuovamente `\clearpage`.

Controllo dello sviluppo dei comandi anche questi ulteriori comandi servono per controllare la tempistica dello sviluppo dei comandi.

Gestione dei *ganci* questi comandi permettono di gestire i "ganci" attaccati ad altri comandi.

Modifica di comandi vale quanto detto sopra per gli strumenti messi a disposizione dei programmatori di classi e di estensioni.

Variabili booleane è questa la parte che ci interessa di più in questo paragrafo e ce ne occuperemo fra un poco.

Test in generale estende i comandi per eseguire test senza fare uso esplicitamente di variabili booleane.

Elaborazione di liste questo è un argomento poco coperto dal linguaggio di LaTeX e gli strumenti di questa sezione erano attesi da tempo.

Miscellanea di strumenti vari vengono aggiunti ancora alcuni strumenti non classificabili nelle categorie precedenti; servono principalmente per lavorare con numeri romani come se fossero numeri in cifre arabe.

La documentazione è ben fatta e succinta; qualunque utente che voglia cimentarsi con la scrittura di macro farebbe bene non solo ad adottare l'uso sistematico di *etoolbox*, ma anche a studiarsi con attenzione la descrizione dei singoli comandi dai quali si può imparare molto.

Il comando `\newbool` non è molto diverso dal comando analogo del pacchetto *ifthen*; la differenza sta nel fatto che esegue una verifica che il nome sia disponibile per una definizione; il comando associato `\providebool` provvede alla definizione solo se la variabile non è ancora definita; Come con il pacchetto *ifthen* sono da considerasi variabili booleane i nomi formati dalle stringhe di caratteri che seguono `\if` in tutti i comandi condizionali definiti con `\newif` e lo sono anche le stringe che seguono la stringa `\if` dei comandi primitivi: per esempio, la variabile booleana `odd` legata al test primitivo `\ifodd`; la variabile booleana `mmode` legata al test primitivo `\ifmmode`.

Questo pacchetto *etoolbox* però esegue i test sul valore della variabile booleana mediante un comando più diretto:

`\ifbool{`⟨*variabile*⟩`}{`⟨*esegui se vero*⟩`}{`⟨*esegui se falso*⟩`}`

Nello stesso modo ci sono comandi più semplici per impostare una variabile a vero o falso, o per impostarla come si farebbe con *ifthen*. Interessante è anche il comando negato:

`\notbool{`⟨*variabile*⟩`}{`⟨*esegui se non vero*⟩`}{`⟨*esegui se non falso*⟩`}`

Utilissimi sono i comandi per verificare se un comando esista già (`\ifdef`) o anche se il nome di un comando corrisponda a un comando esistente (`\ifcsdef`). Bisogna ricordarsi che il nome di un comando è il comando senza il backslash iniziale; ma quando si specifica il nome di un comando lo si può fare anche

senza rispettare le regole che sia formato solo da lettere, oppure solo da un unico carattere non alfabetico. Così esistono `\ifundef` e `\ifcsundef`. Ma esistono anche `\ifdefprefix` e `\ifcsprefix` per sapere se una macro è stata definita con il prefisso `\long` o il prefisso `\protect`; una macro di tipo "long" accetta come argomenti uno o più capoversi, mentre se non ha ricevuto al momento della sua definizione il prefisso `\long` può accettare come argomento solo una stringa che non contenga né direttamente né indirettamente nemmeno una fine di paragrafo. `\protect` caratterizza le macro presumibilmente robuste.

Molto interessanti i comandi `\ifdefempty` e `\ifcsempty` per sapere se il testo di sostituzione di una macro è vuoto; `\ifdefequal` e `\ifcsequal` per sapere se due macro hanno lo stesso numero di argomenti e lo stesso testo sostitutivo oltre agli stessi prefissi; `\ifdefstring` e `\ifcsstring` per sapere se il testo sostitutivo di una macro è uguale ad una stringa di caratteri data.

Comandi analoghi esistono per testare il nome dei contatori TEX (definiti con `\newcount`) o LATEX (definiti con `\newcounter`), e di lunghezze

Poi ci sono test per confrontare stringhe di caratteri indipendenti dalla loro categoria di appartenenza; questi confronti implicano che le due stringhe non siano esplicite entrambe; per esempio si potrebbero usare dentro la definizione di una macro per controllare se un parametro sia vuoto oppure sia costituito da uno spazio; per esempio:

```
\newcommand\prova[1]{%
\iffblank{#1}{esegui se vero}{esegui se falso}}
```

controlla se l'unico parametro passato alla macro quando viene chiamato sia vuoto o sia uno spazio, poi si comporta conseguentemente.

Il confronto fra numeri è gestito con `\ifnumcomp` con la sintassi:

`\ifnumcomp`⟨*test numerico*⟩`{`⟨*esegui se vero*⟩`}{`⟨*esegui se falso*⟩`}`
dove:
⟨*test numerico*⟩
è:
`{`⟨*prima espressione numerica*⟩`}{`⟨*relazione*⟩`}{`⟨*seconda espressione numerica*⟩`}`

Si noti che la relazione è espressa mediante i soliti segni `=`, `>` e `<`. Invece è interessante che ciascuna delle due ⟨*espressioni numeriche*⟩ sia qualunque espressione numerica che abbia senso fra i delimitatori `\numexpr` e `\relax`, senza però contenere questi delimitatori; ci si ricordi che le espressioni numeriche e di lunghezze vengono arrotondate e non troncati per cui un test del tipo:

`\ifnumcomp{6/2}{=}{5/2}{`⟨*esegui se vero*⟩`}{`⟨*esegui se falso*⟩`}`

procede ad eseguire ⟨*esegui se vero*⟩ perché $6/2 = 3$ (e su questo non ci sono dubbi) ma $5/2 = 2{,}5$ per cui le regole dell'arrotondamento portano il risultato finale a 3.

Il comando generale `\ifnumcomp` si semplifica e diventa più immediato da capire se invece di specificare il segno di relazione si usa uno dei tre comandi

`\ifnumequal`, `\ifnumgreater` e `\ifnumless`; il comando primitivo `\ifodd` diventa `\ifnumodd` ma riceve i soliti due argomenti per eseguire cose diverse a seconda che il test sia vero o falso.

Analoghi comandi servono per controllare due espressioni dimensionali.

I comandi per le espressioni logiche sono superficialmente simili nel senso che per singoli valori non sono necessarie le parentesi tonde, ma se queste ci sono servono per alterare l'ordine di esecuzione delle operazioni logiche; in più *etoolbox* consente di gestire ulteriori operatori logici oltre ai soliti `and`, `or` e `not`, si noti però che ora gli operatori *non* sono comandi, ma semplici parole.

Il vantaggio nell'usare le macro di *etoolbox* è che sono molto più numerose e svolgono con semplicità numerose funzioni che richiederebbero scritture più lunghe rispetto a `\ifthen`, ma per lo più si tratta di comandi robusti e quindi possono venire usati senza paura che possano dare luogo a errori di difficile comprensione.

29.9 Spaziature

Gli spazi orizzontali e verticali possono venire inseriti a mano con i seguenti comandi:

```
\hspace{⟨lunghezza⟩}
\hspace*{⟨lunghezza⟩}
\vspace{⟨lunghezza⟩}
\vspace*{⟨lunghezza⟩}
```

I comandi con asterisco impediscono che gli spazi siano eliminati alla fine o all'inizio di una riga oppure all'inizio o alla fine di una pagina, come invece avviene con gli spazi 'semplici'. Questa possibilità di essere eliminati è essenziale per la buona composizione di una pagina e delle sue righe, ma può non essere quello che si desidera quando si compone dentro ad una scatola oppure quando si eseguono composizioni speciali: si pensi per esempio ad un frontespizio dove non si vuole inserire il primo elemento scritto all'inizio della gabbia, ma lo si desidera spostare in basso per l'equilibrio della composizione sulla pagina.

Sono molto comodi i comandi abbreviati

```
\bigskip
\medskip
\smallskip
```

che equivalgono rispettivamente a

```
\vspace{\bigskipamount} \vspace{\medskipamount} \vspace{\smallskipamount}
```

Le lunghezze indicate con i tre comandi `\...skipamount` sono definiti nei file di classe, ma generalmente essi hanno i valori di 12 pt, 6 pt e 3 pt.

Il comando

`\addvspace{`⟨*lunghezza*⟩`}`

aggiunge spazio verticale tenendo conto di eventuale spazio verticale già inserito implicitamente da comandi precedenti, così che alla fine lo spazio complessivo aggiunto ammonti esattamente alla ⟨*lunghezza*⟩ specificata.

I comandi `\hfill` e `\vfill` equivalgono rispettivamente a `\hspace{\fill}` e `\vspace{\fill}`.

29.10 Figure, tabelle ed altri oggetti flottanti

I comandi per rendere flottanti le figure e le tabelle sono descritti nei prossimi paragrafi.

29.10.1 Figure e tabelle

`\begin{figure}[`⟨*posizione*⟩`]` ⟨*figura*⟩ `\end{figure}`
`\begin{figure*}[`⟨*posizione*⟩`]` ⟨*figura*⟩ `\end{figure*}`
`\begin{table}[`⟨*posizione*⟩`]` ⟨*tabella*⟩ `\end{table}`
`\begin{table*}[`⟨*posizione*⟩`]` ⟨*tabella*⟩ `\end{table*}`

I comandi senza asterisco vanno bene sia componendo ad una colonna, sia componendo a due colonne. I comandi con l'asterisco inseriscono l'oggetto a larghezza piena in testa ad una pagina composta a due colonne; ma componendo ad una colonna si può usare indifferentemente la versione con o senza asterisco.

Le posizioni possibili sono

`t`	in testa alla pagina
`b`	alla base della pagina
`h`	'qui', se possibile...
`p`	in una pagina di soli oggetti flottanti

Con tutte le classi standard, se l'argomento facoltativo ⟨*posizione*⟩ non è specificato, le posizioni di default sono `tbp`.

La lista delle posizioni in realtà accetta anche `!` che invita LaTeX a 'mettercela tutta' per collocare l'oggetto anche trascurando alcuni dei parametri di posizionamento dettati dalla classe in uso.

LaTeX può mettere un filetto sotto a una figura in testa alla pagina, oppure sopra una figura in calce alla pagina; può mettere anche un filetto sotto ad una figura a piena pagina in testa ad una pagina composta a due colonne. I comandi per mettere questi filetti sono `\topfigrule`, `\botfigrule` e `\dblfigrule`; di default questi tre comandi sono equivalenti a `\relax`, quindi non fanno niente, Ogni classe oppure ogni compositore li può ridefinire in modo che mettano

qualcosa, generalmente un filetto, ma potrebbe essere una qualunque decorazione, nella posizione a cui si riferiscono; però questo qualcosa deve avere altezza *nulla* allo stesso modo del filetto che separa le note in calce dal testo sovrastante. La definizione di questo filetto `\footnoterule` è la seguente:

```
\def\footnoterule{\kern-3\p@ \hrule \@width 2in \kern 2.6\p@}
```

e basta definire i comandi di questi filetti cambiando il nome `\footnoterule` con `\topfigrule`, o `\botfigrule`, o `\dblfigrule`; per esempio:

```
\def\topfigrule{\kern2.6\p@ \hrule \kern-3\p@}
\def\botfigrule{\kern-3\p@ \hrule \kern 2.6\p@}
\def\dblfigrule{\kern2.6\p@ \hrule \kern-3\p@}
```

Si noti l'espediente per fare sembrare il filetto di altezza nulla; si arretra di 3pt, si mette un filetto spesso 0,4pt, e si avanza di 2,6pt; complessivamente non si è avanzato in verticale di nessuno spazio. Gli spostamenti in avanti o all'indietro di 2,6pt e di 3pt sono messi strategicamente in modo che lo spostamento positivo separi il filetto da ciò che esso segue o precede.

LaTeX, durante la costruzione della pagina di uscita, segue dei criteri di ottimalità per collocare gli oggetti flottanti che eventualmente sono accodati per essere emessi appena possibile. Questi criteri implicano il rispetto di certe frazioni minime o massime di spazio dedicato a questa o quella parte del documento, nonché al numero massimo di oggetti che possono essere collocati in una pagina. Le regole che LaTeX segue sono descritte qui di seguito, ma è chiaro che se si vuole che LaTeX faccia diversamente da come è stato programmato, bisogna comprendere appieno queste regole e bisogna capire come funzionano i parametri massimi o minimi che vi presiedono.

1. LaTeX colloca un oggetto flottante nel primo posto che non viola le regole seguenti, salvo che, se è specificato il codice `h`, questo ha la precedenza sul codice `t`.
2. LaTeX non collocherà mai un oggetto flottante in una pagina che viene prima di quella dove compare il testo del file sorgente che attornia l'ambiente di flottaggio.
3. LaTeX colloca le figure in ordine e lo stesso fa con le tabelle, per cui non può succedere che la figura 22 appaia prima della figura 21.
   Va notato però che quando si compone a due colonne (senza usare il pacchetto *multicol*[6], ma usando la specifica LaTeX `\twocolumn`) possono formarsi pagine che hanno figure o tabelle a giustezza piena contemporaneamente a una o più figure o tabelle con la giustezza di una colonna. In questi casi, visto che gli oggetti a piena giustezza vengono collocate solo in testa, mentre gli oggetti con la giustezza di una colonna possono essere collocati sia in testa alla colonna, sia in calce, sia in una posizione qualsiasi consentita dai parametri di posizione, potrebbe succedere che

[6] Questo pacchetto non consente di mettere figure all'interno delle colonne.

l'oggetto in testa abbia un numero successivo a quello di un oggetto in colonna. Questo è quanto succedeva in effetti fino al 2015, per cui era necessario ovviare a questo inconveniente usando un piccolo pacchetto di correzioni del nucleo di LaTeX, *fixltx2e*, che eseguiva alcune modifiche alla routine di emissione della pagina composta, onde evitare questa e poche altre apparenti incongruenze presenti nel nucleo standard. La documentazione di questo pacchetto si trova in `.../doc/latex/base/`, ma l'utente generalmente dovrebbe usare questo pacchetto solo quando compone a due colonne e se ha numerose figure o tabelle di varie dimensioni orizzontali, altrimenti questi difettucci non si manifestano quando si compone ad una sola colonna. Dal 2016 queste modifiche sono integrate nel nucleo di LaTeX, per cui non è più necessario ricorrere a *fixltx2e*.

4. LaTeX colloca gli oggetti flottanti solamente nelle posizioni specificate nell'argomento facoltativo ⟨*posizione*⟩ o, in mancanza di questo, in una delle tre posizioni di default `tbp`.
5. LaTeX non collocherà mai un oggetto flottante in una pagina che non contiene abbastanza spazio, quindi non produrrà mai una '`Overfull vbox`' a causa degli oggetti flottanti, almeno se un oggetto flottante di per sé non è così grande da eccedere lo spazio disponibile; a questo l'utente deve provvedere preventivamente mediante le opportune operazioni di scalamento e gli altri artifici descritti negli appositi capitoli relativi a figure e tabelle.
6. I vincoli imposti dai parametri stilistici tipici di ogni classe non vengono mai violati; quando viene specificato `!` vengono rilassati i vincoli imposti dai parametri che contengono sia testo sia oggetti flottanti, mentre LaTeX continua a rispettare i vincoli imposti alle pagine che contengono solamente oggetti flottanti. Per queste pagine di soli oggetti flottanti le regole vengono ignorate quando si emettono i comandi `\clearpage` oppure `\cleardoublepage`, e, ovviamente, quando si incontra `\end{document}`, perché questi comandi e la specifica che il documento è terminato ordinano a LaTeX di svuotare le code di tutto ciò che si trova ancora in memoria.
7. Non è una buona idea specificare una sola posizione per un oggetto flottante, in particolare non è opportuno specificare solo `h`. Se l'oggetto non può stare proprio lì dove compare il suo codice sorgente, viene messo in coda e viene trattato come un oggetto con la specificazione `t`. Tuttavia questo, pur rendendo può facile il posizionamento dell'oggetto flottante in una pagina successiva a quella desiderata, subisce lo stesso trattamento degli altri oggetti eventualmente in coda, quindi potrebbe apparire dopo molte pagine rispetto a quella desiderata. Inoltre se è troppo alto anche per un oggetto di tipo `t`, resta in coda fino alla fine di un capitolo o nel peggiore dei casi fino alla fine del documento; così facendo esso blocca la coda e impedisce il deflusso anche degli altri oggetti.

Quando si specifica la ⟨*posizione*⟩ bisogna dare abbastanza possibilità a LaTeX di fare il suo mestiere, altrimenti l'oggetto che non ha altre possibilità

che una sola, può bloccare le code fino alla fine del documento o alla prima esecuzione di `\clearpage` o `\cleardoublepage`; questi comandi vengono emessi automaticamente quando si inizia un nuovo capitolo, ma, a parte che è brutto vedere (quasi) tutte le figure di un capitolo accumulate alla sua fine, bisogna ricordare che la memoria di pdflatex è limitata; dal 2017 è programmata per memorizzare 52 oggetti flottanti (fino al 2016 questo numero valeva 18); se dovessero accodarsene di più, quelli in eccesso verrebbero persi e verrebbe emesso un messaggio che avvisa del fatto, ma è una magra consolazione. Si esamini anche la possibilità di usare il pacchetto *morefloats*. Con lualatex e xelatex il numero degli oggetti flottanti in coda è sempre potuta arrivare fino a 52.[7]

Solo pdflatex

Per lualatex e xelatex

I comandi che si possono usare dentro gli ambienti di flottaggio tipicamente sono:

`\caption` con la sintassi

```
\caption[⟨didascalia breve⟩]{⟨didascalia⟩}
```

dove se non si specifica la ⟨*didascalia breve*⟩ questa viene automaticamente resa identica alla didascalia 'lunga'; potrebbe non essere una buona idea quella di inviare alla lista delle figure o delle tabelle l'intera didascalia, specialmente se questa contiene uno o più periodi dopo il primo che svolge il compito di titolo, mentre i periodi successivi svolgono il compito di fornire maggiori delucidazioni sull'oggetto specifico.

`\label` può essere usato, anzi è conveniente che lo sia, per ogni figura e per ogni tabella; il comando deve essere collocato dopo il comando `\caption`, perché finché questo non viene eseguito, all'oggetto flottante non è ancora stato assegnato un numero.

`\suppressfloats` con la sintassi

```
\suppressfloats[⟨posizione⟩]
```

può essere collocato fuori degli ambienti di flottaggio al fine di evitare che il programma metta altri oggetti nella stessa ⟨*posizione*⟩ specificata come argomento facoltativo; se non si specifica nulla, tutti gli oggetti flottanti sono esclusi dalla pagina corrente; se però nella ⟨*posizione*⟩ del comando di apertura degli ambienti *figure* o *table* compare `!`, allora `\suppressfloats` viene ignorato.

I parametri dimensionali e numerici che regolano il deflusso degli oggetti flottanti dalle rispettive code sono i seguenti.

`topnumber` è il nome del contatore che contiene il numero *massimo* di oggetti flottanti che possono essere collocati in testa alla pagina.

[7] Dal 2020 la gestione degli oggetti flottanti è cambiata in parte; la modifica maggiore consiste nel fatto che ora il numero massimo di oggetti flottanti è talmente grande, memoria di lavoro permettendo, che è lecito non preoccuparsi più dell'esistenza dei limiti precedentemente descritti.

`\topfraction` è la *massima* frazione di pagina destinata agli oggetti flottanti in testa ad una pagina, se questa contiene anche del testo.

`bottomnumber` è il contatore che contiene il numero *massimo* di oggetti flottanti in calce alla pagina.

`\bottomfraction` è la *massima* frazione di pagina destinata agli oggetti flottanti in calce ad una pagina, se questa contiene anche del testo.

`totalnumber` è il contatore che contiene il *massimo* numero di oggetti flottanti che possono comparire in una pagina di testo, indipendentemente dal fatto che siano in testa o in calce o in mezzo al testo.

`\textfraction` per una pagina che contenga anche del testo, è la frazione *minima* della pagina destinata al testo.

`\floatpagefraction` in una pagina di soli oggetti flottanti rappresenta la *minima* frazione di pagina che deve essere occupata da questi oggetti; se cioè si specifica il parametro di posizione `p` per un oggetto flottante e questo è troppo piccolo, esso viene trattenuto in memoria finché non si trova un altro oggetto flottante dello stesso genere e con lo stesso attributo di posizione che assieme possano superare questo valore minimo specificato.

`dbltopnumber` il nome del contatore che contiene il numero *massimo* di oggetti flottanti a piena pagina da mettere in testa ad una pagina con il testo composto su due colonne.

`\dbltopfraction` la *massima* frazione di pagina a giustezza piena da destinare agli oggetti flottanti in testa alla pagina quando si compone a due colonne.

`\dblfloatpagefraction` è la *minima* frazione di pagina da occupare con oggetti flottanti a giustezza piena quando si compone una pagina di soli oggetti flottanti in un testo composto a due colonne. È, insomma, l'analogo di `\floatpagefraction` che, invece, vale quando si compone il testo ad una sola colonna.

`\floatsep` è la distanza *minima* da riempire con un contrografismo incolore fra due oggetti flottanti consecutivi.

`\textfloatsep` è la distanza *minima* da interporre fra gli oggetti flottanti in testa o in calce e il testo adiacente.

`\intextsep` è la distanza *minima* fra un oggetto flottante a centro pagina e il testo che lo precede e quello che lo segue.

`\dblfloatsep` è la distanza *minima* fra due oggetti flottanti consecutivi in testa ad una pagina composta a due colonne.

`\dbltextfloatsep` è la distanza *minima* posta fra gli oggetti flottanti in testa ad una pagina composta a due colonne e il testo sottostante.

I contatori sono contatori LaTeX quindi si impostano con i comandi specifici, in particolare `\setcounter`; le frazioni `\...fraction` sono valori decimali, in generale fratti e minori dell'unità che vengono conservati dentro le macro con i rispettivi nomi, quindi, per modificarne i valori, tutti si devono reimpostare mediante `\renewcommand`. I separatori `\...sep` sono lunghezze e si impostano con il comando `\setlength`. Ogni classe definisce i suoi particolari valori per questi parametri; per la classe *`book`* composta in corpo 10 (come questo testo) i

Numeri		Frazioni	
`topnumber`	2	`\topfraction`	0,7
`bottomnumber`	1	`\bottomfraction`	0,3
`totalnumber`	3	`\textfraction`	0,2
`dbltopnumber`	2	`\floatpagefraction`	0,5
		`\dbltopfraction`	0,7
		`\dblfloatpagefraction`	0,5

Separatori	
`\floatsep`	`12pt plus 2pt minus 2pt`
`\textfloatsep`	`20pt plus 2pt minus 4pt`
`\intextsep`	`12pt plus 2pt minus 2pt`
`\dblfloatsep`	`12pt plus 2pt minus 2pt`
`\dbltextfloatsep`	`20pt plus 2pt minus 4pt`

Tabella 29.3: Parametri nella classe *book* composta in corpo 10 per gestire gli oggetti flottanti

parametri sono riportati nella tabella 29.3.

Va detto che i 'Numeri' e le 'Frazioni' indicati nelle tabelle 29.3 non devono necessariamente essere coerenti; sembra strano, ma è comprensibile. Per i 'Numeri', per esempio, `totalnumber` sembra essere la somma di `topnumber` e `bottomnumber`; se però `totalnumber` fosse posto al valore 2, questo vorrebbe dire che *al massimo* ci possono essere due oggetti flottanti in testa e uno in calce, ma in ogni caso non possono essere tutti e tre presenti, perché nella pagina ce ne possono essere *al massimo* due. Lo stesso vale per le frazioni; esse rappresentano dei limiti per delle disuguaglianze; questi limiti non debbono necessariamente avere per somma l'unità; l'importante è che queste disuguaglianze siano tutte rispettate nell'ordine di preferenza che LaTeX assegna alla collocazione degli oggetti flottanti.

Il compositore si ricordi, specialmente per quel che riguarda le frazioni, che nel computo dell'ingombro dell'oggetto flottante bisogna tenere conto anche dello spazio destinato alla didascalia e del suo spazio di separazione. Spesso il compositore è deluso dalla rigidezza con cui il programma gestisce gli oggetti flottanti; in realtà il programma, come tutti i programmi, si comporta come gli è stato prescritto, in particolare, in questo caso, come richiesto dai 'Numeri', dalle 'Frazioni' e dai 'Separatori' che compaiono nella tabella 29.3. Se non piacciono quei valori, li si può cambiare, ma il compositore stia attento che la cura non sia peggiore del male. Piuttosto egli dedichi la sua attenzione al dimensionamento degli oggetti flottanti, all'eliminazione di spazi inutili al loro contorno, all'eliminazione delle parti inutili delle fotografie che si trovano spessissimo ai bordi laterali o verticali; insomma usi appropriatamente le possibilità offerte dalle opzioni di

`\includegraphics` perché è con quelle che si curano davvero sia il deflusso degli oggetti flottanti dalle rispettive code, sia la qualità del documento prodotto.

Analogamente per le tabelle: le si progetti in anticipo con grande cura e non si pretenda l'impossibile. È successo che sul forum del GUIT venissero presentate delle 'lamentele' perché pdflatex non metteva questa o quella tabella dove il richiedente la voleva mettere; avendo richiesto un esempio minimo compilabile, veniva presentata una tabella con lo spazio attorno al contenuto di ciascuna cella, che di default vale 12 pt (6 pt a sinistra e 6 pt a destra), cioè in totale circa 4 mm, da solo, tenuto conto della quindicina di colonne portava la larghezza totale a circa 60 mm; ogni cella era in colonne di larghezza specificata e non inferiore a 25 mm; si fa presto a fare il conto; ogni cella occupava non meno di circa 29 mm che con 15 celle in una riga della tabella porta il totale a 435 mm. Ovviamente una tale tabella non sta né in orizzontale né in verticale in nessuna pagina di formato A4. Questo errore è plateale, ma indica che spesso le tabelle non vengono progettate per niente.

29.10.2 Note marginali

I comandi che portano alla composizione e alla collocazione delle note marginali sono i seguenti:

```
\marginpar[⟨nota a sinistra del testo⟩]{⟨nota a destra del testo⟩}
\reversemarginpar
\normalmarginpar
```

La posizione normale delle note quando si compone ad una colonna è il margine esterno; quando si compone a due colonne è il margine adiacente alla colonna alla quale la nota si riferisce, quindi il margine di sinistra per la colonna di sinistra e il margine di destra per la colonna di destra; e questo avviene indipendentemente dal fatto che ci si trovi in una pagina di sinistra o di destra. `\reversemarginpar` scambia i margini per le note composte ai margini di testi composti ad una sola colonna e `\normalmarginpar` ripristina la collocazione di default. Questi due comandi sono inattivi quando si compone a due colonne.

Per il testo delle note che dovrebbero apparire a sinistra del testo a cui si riferiscono potrebbe essere applicata la dichiarazione `\raggedleft` per avere una composizione in bandiera giustificata a destra. Tuttavia questo è consigliabile solo quando le annotazioni marginali sono sempre molto brevi in modo da non superare una riga. Cambiare giustificazione da nota a nota sarebbe una caduta di uniformità e di stile. D'altra parte le note di più righe giustificate solo a destra sono leggermente più laboriose da leggere rispetto alle note giustificate solo a sinistra; il compositore ci pensi bene prima di decidere come comporre le note; eventualmente si crei una macro che componga sempre entrambi i testi di sinistra e di destra uniformemente con lo stesso stile, in modo da non dover operare a mano così da evitare i possibili errori di disomogeneità.

I parametri che governano la collocazione delle note, a parte il margine 'normale' o 'scambiato', sono:

\marginparwidth è la giustezza delle note marginali e deve essere evidentemente minore della larghezza del margine fisico a lato del testo.

\marginparsep è lo spazio di separazione fra il testo e il blocchetto della nota marginale.

\marginparpush è lo spazio con cui una nota marginale viene spostata in basso rispetto alla fine della nota marginale precedente; talvolta questo obbliga a portare la nota nella pagina successiva, ma viene emesso un avvertimento sullo schermo e nel file `.log`. Per riparare queste situazioni non tanto desiderabili, bisogna necessariamente riformulare il testo principale a cui la nota in questione si riferisce, oppure, bisogna accorciare il testo della nota precedente.

Un avvertimento: il programma tratta le note marginali alla stessa stregua degli oggetti flottanti; questo vuol dire che usa le stesse scatole per conservare in memoria gli oggetti flottanti; siccome le scatole a disposizione erano limitate, l'uso di frequenti note marginali e di molte figure e/o tabelle aumentava il rischio di esaurire la memoria a disposizione e di perdere degli oggetti flottanti.

Solo pdflatex

Si poteva fare uso del pacchetto *morefloats* che raddoppia il numero di oggetti flottanti in coda; permetteva di arrivare fino a 36. Il motivo per il quale esso arrivava solo fino a una coda di 36 oggetti flottanti era per evitare di esaurire la *main memory* del programma di composizione. Tuttavia va osservato che il pacchetto fu scritto nel 1990, quindi quando la memoria a disposizione dei vari main frame e dei PC personali era veramente piccola. Il sistema TeX che sto usando dispone di una memoria di 3 000 000 parole di 4 byte per svolgere il suo lavoro, ma ne usa solamente la settima parte, come si può leggere alla fine del file `.log`. Invece con lualatex e xelatex la gestione della memoria è diversa e si tratta di programmi resi disponibili in tempi recenti, cosicché non hanno più bisogno di limitazioni così stringenti sul numero di oggetti in coda; di per sé i programmi avevano una limitazione a 52 oggetti in coda, quindi significativamente maggiore di quello che si poteva ottenere con il pacchetto *morefloats*, tanto che questo non ha più bisogno di essere usato con questi programmi; si veda il capitolo 30 dove si spiega come il limite sul numero di oggetti flottanti in coda ora è grandissimo per tutti i programmi di composizione.

Per lualatex e xelatex

29.11 Incolonnamenti

29.11.1 L'ambiente *tabbing*

Nel testo non se ne è nemmeno parlato, perché si ritiene che questo ambiente sia una reminiscenza degli incolonnamenti eseguiti in dattilografia; quando TeX 78 nacque, le macchine da scrivere meccaniche ed elettriche erano ancora diffusissime e le tabulazioni eseguite con il tasto di tabulazione, nonché con gli arresti di

tabulazione, erano familiari a tutti. Tuttavia l'ambiente *tabbing*, nato, forse, proprio per affidarsi a qualcosa di noto a qualunque dattilografo, presenta dei notevoli inconvenienti, primo fra i quali quello di essere un ambiente fragile e che non può essere annidato dentro altri ambienti.

La sintassi è quella comune a qualunque ambiente:

`\begin{tabbing}` ⟨*testo da incolonnare*⟩ `\end{tabbing}`

I comandi per gestire gli arresti di tabulazione sono i seguenti.

\= serve per impostare un arresto di tabulazione; spesso conviene scrivere una riga modello da *non* comporre se si ricorre al comando seguente.

\kill non compone la riga che precede questa parola chiave, ma conserva le informazioni di impostazione degli arresti di tabulazione che la riga conteneva.

\> serve per spostare la composizione al successivo arresto di tabulazione.

**** comincia una nuova riga riportando il contatore degli arresti di tabulazione al valore iniziale, generalmente zero.

\+ incrementa di una unità il valore iniziale del contatore di tabulazione così che da questo momento in poi e fino ad ordine contrario, tutte le righe risultano rientrate e incolonnate sotto un arresto presente nella riga precedente.

\- decrementa di una unità il valore iniziale del contatore di tabulazione; contrasta l'effetto di `\+`.

\< torna indietro di un arresto di tabulazione e può essere usato solamente all'inizio di una riga.

\' serve per spostare tutto il contenuto di una 'colonna' a filo del margine destro del suo campo di tabulazione.

\` serve per comporre il contenuto della sua colonna a filo del margine sinistro del suo arresto di tabulazione.

\pushtabs manda in memoria le posizioni degli attuali arresti di tabulazione, consentendo di impostarne di diversi.

\poptabs serve per richiamare dalla memoria un insieme di arresti di tabulazione precedentemente memorizzati.

\a= `\a'` e `\a`` agiscono come `\=`, `\'` e `\`` per comporre gli accenti macron (lungo), acuto e grave rispettivamente, perché questi comandi, come si vede sopra, sono stati ridefiniti all'interno dell'ambiente per svolgere funzioni diverse. Il compositore che usi i caratteri nazionali della sua tastiera o immetta direttamente caratteri con la codifica UTF-8 non se ne preoccupi e continui ad usare tranquillamente le lettere accentate della sua tastiera, perché si provvede automaticamente alla compatibilità senza che il compositore debba preoccuparsene più di tanto. Il compositore deve purtroppo occuparsene se e solo se la sua tastiera è priva di segni accentati e se il suo *shell editor* non consente di usare combinazioni di tasti per inserire direttamente nel file `.tex` i segni accentati.

29.11.2 Gli ambienti *array* e *tabular*

Le sintassi sono:

```
\begin{array}[⟨allineamento⟩]{⟨colonne⟩}
    ⟨righe⟩
\end{array}
```

```
\begin{tabular}[⟨allineamento⟩]{⟨colonne⟩}
    ⟨righe⟩
\end{tabular}
```

```
\begin{tabular*}{⟨larghezza⟩}[⟨allineamento⟩]{⟨colonne⟩}
    ⟨righe⟩
\end{tabular*}
```

L'⟨*allineamento*⟩ serve per stabilire la posizione verticale della tabella o della matrice con il 'testo' circostante; la ⟨*larghezza*⟩ serve per imporre una larghezza specificata alla tabella. I descrittori delle ⟨*colonne*⟩ sono certi codici che ora si rivedranno; le ⟨*righe*⟩ sono i contenuti delle celle orizzontali contenenti il testo o la matematica da comporre. *array* si usa solo in ambiente matematico; gli ambienti *tabular* e *tabular** solo in modo testo; sia gli uni sia gli altri ambienti possono contenere singole celle contenenti 'testo' dell'altra specie, ma bisogna specificarlo per ciascuna di queste celle; in ambiente *tabular* basta usare gli appositi comandi per passare alla matematica; in ambiente *array* si possono usare i segni di dollaro che agiscono da interruttori/deviatori; se si è già in modo matematico (come avviene per ogni cella di un *array*) il primo segno di dollaro passa al modo testo e il secondo riporta al modo matematico.

I codici di allineamento sono:

`t` serve per allineare la riga di testa con il testo circostante.
`b` serve per allineare la riga di base con quella del testo circostante.
`c` non necessita di essere espresso, perché l'allineamento centrato è quello di default.

I descrittori delle ⟨*colonne*⟩ e i loro separatori sono i seguenti.

`l` colonna con i contenuti allineati a sinistra.

`c` colonna con i contenuti centrati.

`r` colonna con i contenuti allineati a destra.

`|` filetto verticale di separazione fra colonne adiacenti.

`\vline` lo stesso filetto quando deve essere inserito in una *@-espressione*.

`@{`⟨*testo*⟩`}` *@-espressione*; si tratta di un particolare costrutto con il quale si specifica che ⟨*testo*⟩ sostituisce completamente il separatore ordinario fra due colonne adiacenti. Questo significa che se i contenuti delle due celle adiacenti devono essere in qualche modo distanziati, allora gli spaziatori devono essere inseriti dentro la *@-espressione*. Un elemento importante delle *@-espressioni* sono le dichiarazioni di spaziatura con elasticità 'infinita' da usare con l'ambiente *tabular** per consentire che queste particolari tabelle si possano estendere fino a raggiungere la larghezza specificata.

`\extracolsep{`⟨*larghezza*⟩`}` specificato dentro una *@-espressione* serve per dichiarare che lo spaziatore con la ⟨*larghezza*⟩ esplicitata venga inserito a sinistra del contenuto di tutte le celle *che seguiranno* sulla stessa riga. Tipicamente questo comando viene inserito nella prima *@-espressione* che compare a sinistra della prima cella di una riga; quindi questa prima cella non verrà mai allargata per raggiungere la giustezza desiderata.

`\fill` è una larghezza naturalmente nulla ma dotata di allungamento infinito, tipicamente usata come argomento di `\extracolsep`.

`p{`⟨*larghezza*⟩`}` serve per descrivere una colonna composta come un (breve) capoverso la cui prima riga è allineata con quella delle celle adiacenti e avente una giustezza pari a ⟨*larghezza*⟩. Il testo delle colonne composte con questo descrittore non può contenere direttamente il comando `\\` perché LaTeX non potrebbe sapere se esso si riferisce alla cella o all'intera riga di celle. Perciò è necessario racchiudere questo comando dentro un ambiente, come *minipage* o un altro *tabular*, oppure dentro una scatola `\parbox` esplicita; oppure dentro il raggio di azione di comandi come `\centering`, `\raggedright` oppure `\raggedleft`, purché a loro volta siano contenuti dentro un gruppo delimitato da parentesi graffe oppure siano contenuti all'interno di altri ambienti.

`*{`⟨*numero*⟩`}{`⟨*descrittori*⟩`}` è una forma abbreviata per ripetere ⟨*numero*⟩ volte le stessa sequenza di ⟨*descrittori*⟩ delle colonne. Una `*`-*espressione* ne può contenere un'altra; si consiglia di non eccedere con queste forme stenografiche innestate l'una nell'altra altrimenti dopo un po' non si capisce più che cosa si sia specificato. Piuttosto è meglio rifarsi alle possibilità offerte dal pacchetto *array* e dal suo comando `\newcolumntype`.

I comandi che si possono usare dentro gli ambienti di incolonnamento sono i seguenti.

`\multicolumn` con la sintassi

```
\multicolumn{⟨numero⟩}{⟨descrittore⟩}{⟨cella⟩}
```

serve per comporre un'unica ⟨*cella*⟩ con il ⟨*descrittore*⟩ specificato che occupi ⟨*numero*⟩ celle adiacenti. Il ⟨*descrittore*⟩ oltre alla collocazione con

una delle varie lettere chiave (compresa la `p`) può contenere filetti verticali e persino *@-espressioni*. Normalmente il contenuto di `\multicolumn` rimpiazza completamente tutte le ⟨*numero*⟩ celle adiacenti, compresi i loro separatori destri; quindi il separatore sinistro non dovrebbe mai essere indicato come contenuto del ⟨*descrittore*⟩ a meno che il gruppo di celle non comprenda anche la prima cella di una riga.

`\hline` serve per tirare un filetto orizzontale attraverso tutta la tabella o tutta la matrice. Esso può essere eseguito solo come primo elemento di una tabella, oppure dopo il comando di fine riga `\\`, oppure dopo un altro comando `\hline`; in quest'ultimo caso vengono tracciati due filetti ravvicinati attraverso tutta la tabella. Questo è ciò che fa il linguaggio LaTeX se non si usano file di estensione; usando il file *`booktabs`*, con il quale di fatto non si possono usare filetti verticali, ci sono i comandi `\toprule` da inserire prima della prima riga di celle, `\bottomrule` da inserire alla fine della tabella, e `\midrule` da inserire generalmente solo dopo la prima riga contenente i titoli delle colonne. I primi due filetti sono leggermente più spessi del terzo, ma tutti sono adeguatamente distanziati dal testo loro adiacente, mentre i filetti tracciati con `\hline`, oltre ad essere tutti dello stesso spessore, spesso sfiorano superiormente da vicino le lettere maiuscole, e inferiormente i discendenti delle lettere minuscole; questo inconveniente si può evitare con adeguati trucchetti, ma il file di estensione *`booktabs`* evita di doversene preoccupare.

`\cline{`⟨col_1-col_2⟩`}` serve per tracciare un filetto orizzontale solo sotto le colonne ⟨col_1-col_2⟩; per esempio in una tabella a 7 colonne, il comando `\cline{2-6}` traccia un filetto orizzontale sotto le 5 colonne centrali; non si possono inserire due identici comandi `\cline` uno di seguito all'altro per raddoppiare il filetto, ma si possono usare per 'sottolineare' colonne distinte: per esempio, nella tabella a 7 colonne dell'esempio precedente `\cline{2-3}\cline{5-6}` 'sottolinea' solo le colonne 2 e 3 con un filetto e, allineato con il primo, un secondo filetto 'sottolinea' le colonne 5 e 6.

I parametri compositivi di tabelle e matrici sono i seguenti.

`\arraycolsep` è metà dello spazio di separazione fra due celle consecutive di una matrice.

`\tabcolsep` e metà dello spazio di separazione fra due celle consecutive di una tabella.

`\arrayrulewidth` è lo spessore dei filetti orizzontali o verticali che si possono inserire in tabelle e matrici.

`\doublerulesep` è la spaziatura verticale o orizzontale fra due filetti consecutivi orizzontali o verticali.

`\arraystretch` è una macro, che può venire ridefinita con `\renewcommand`, e che rappresenta il fattore di scala con cui viene ingrandito o rimpicciolito il pilastrino[8] che si trova sempre nella prima cella di ogni riga delle tabelle

[8] In inglese si chiama *strut* da cui discende il nome del comando `\strut`.

o delle matrici. Il suo valore di default è l'unità.

29.12 I file ausiliari e i loro comandi

29.12.1 I file del sistema TEX

LATEX, come anche Plain TEX, fa riferimento ad un gran numero di file dei quali spesso il compositore non si accorge nemmeno. A parte i file sorgente con estensione `.tex` (o `.ltx`), ci sono anche i file con le estensioni seguenti; il nome proprio dei file corrispondenti ad un medesimo documento è costantemente il nome proprio del file principale e questo nome è conservato dentro la variabile `\jobname` che può essere usata anche dal compositore, o meglio, dal programmatore che scrive le macro.

`.aux` serve per conservare tutte le informazioni che riguardano i riferimenti incrociati; quando si usano i comandi `\include`, questi file conservano anche tutte le informazioni relative al file incluso in modo da poterne simulare la compilazione quando questo file non sia incluso nella lista di file da compilare contenuta nell'argomento di `\includeonly`. Il comando `\nofiles` sopprime la scrittura di tutti i file di questo tipo.

`.bbl` questo file viene scritto da BIBTEX e da biber, ma viene poi letto dal programma per comporre la bibliografia quando si usi il comando `\bibliography` o `\printbibliography`.

`.dvi` è il formato di default dell'uscita compilata di tutti i programmi elaborati con il sistema TEX fino a metà degli anni '90; esso fa riferimento ai font a matrici di pixel e generalmente il programma per rendere leggibile agli umani questo file è incluso nella distribuzione del sistema TEX; pdflatex, che viene usato per produrre direttamente l'uscita in formato PDF, è in grado di produrre l'uscita in formato DVI se si pone all'inizio del file `.tex` la specificazione: `\pdfoutput=0`.

`.glo` è il file prodotto da LATEX quando sia attivo il comando `\makeglossary`; esso contiene tutte le voci `\glossaryentry` prodotte dai vari comandi `\glossary` inseriti nei file sorgente.

`.idx` è il file prodotto da LATEX quando sia attivo il comando `\makeindex`; esso contiene tutte le voci `\indexentry` prodotte dai vari comandi `\index` inseriti nel file sorgente.

`.ind` questo file viene scritto dal programma makeindex ma viene riletto da LATEX quando deve comporre l'indice analitico.

`.lof` contiene le informazioni per comporre l'indice delle figure.

`.log` contiene tutta le registrazione di quanto è successo durante l'esecuzione di LATEX, in particolare le informazioni relativamente dettagliate concernenti gli errori e gli avvertimenti, i font usati, quelli che sono stati creati al volo, quelli mancanti, quelli sostituiti, eccetera.

`.lot` contiene le informazioni per comporre la lista delle tabelle.

`.toc` contiene le informazioni per comporre l'indice generale.

Usando certi pacchetti di estensione, come per esempio *hyperref* si generano altri file ausiliari con altre estensioni.

29.12.2 I riferimenti incrociati

I comandi per assegnare una parola chiave agli oggetti da citare e/o per richiamarne il valore o la pagina sono:

```
\label{⟨chiave⟩}
\ref{⟨chiave⟩}
\pageref{⟨chiave⟩}
```

Il comando `\label` era fragile; sarebbe meglio, anche se è possibile, non inserirlo mai all'interno di altri comandi, nemmeno i comandi di sezionamento o i comandi per le didascalie. I contatori che possono essere citati con la chiave sono quelli che sono stati incrementati con `\refstepcounter`; in ogni caso `\label` associa alla ⟨*chiave*⟩ il valore dell'ultimo contatore in ordine di tempo che è stato incrementato con quel comando.

29.12.3 Bibliografia e citazioni

Salvo disporre di file bibliografici esterni (database bibliografici) e di fare uso dei programmi BibTeX o biber, LaTeX dispone dell'ambiente *thebibliography* per comporre gli elenchi bibliografici e assegnare loro una chiave di riferimento.

```
\begin{thebibliography}{⟨stringa⟩}
\bibitem[⟨etichetta⟩]{⟨chiave⟩} ⟨voce bibliografica⟩
...
\end{thebibliography}
```

Le voci bibliografiche vengono richiamate con il comando

```
\cite[⟨testo⟩]{⟨chiave⟩}
oppure
\cite{⟨lista di chiavi⟩}
```

La ⟨*chiave*⟩ è una informazione interna di LaTeX che serve a LaTeX stesso e al compositore per riferirsi ad una particolare voce bibliografica. Quando questa viene citata, viene citata o con l'⟨*etichetta*⟩, se è stata specificata, oppure con il numero progressivo dell'elenco bibliografico; in ogni caso numero o etichetta vengono racchiusi entro parentesi quadre. La ⟨*stringa*⟩ è solo una stringa di segni alfanumerici, spesso una sequenza di cifre, che serve al programma per prendere le misure da assegnare al campo dell'⟨*etichetta*⟩ alfanumerica o puramente numerica con cui identificare ogni singola opera nella bibliografia composta e nelle citazioni.

Con `\cite` si possono fare citazioni multiple, semplicemente specificando una ⟨*lista di chiavi*⟩ formata dalle chiavi specifiche dei riferimenti da citare separate da

virgole; per questo motivo le chiavi possono essere formate con qualunque carattere, esclusa la virgola. Se si cita un'opera sola, la citazione contiene l'⟨*etichetta*⟩, o il numero, seguiti da ⟨*testo*⟩; per esempio: avendo attribuito ad un riferimento l'etichetta `TUG2007`, il comando di citazione `\cite[cap.~2]{TUG2007}` produrrà qualcosa come [12, cap. 2].

Per la bibliografia, specialmente se contiene numerose voci, è consigliabile servirsi dei programmi BibTeX o biber, che garantiscono la composizione uniforme delle varie citazioni e, tramite file di estensione, consentono anche di usare stili di citazione diversi come, per esempio, lo stile 'autore-anno'.

29.12.4 Suddivisione del file sorgente

Non è conveniente scrivere un unico file sorgente, specialmente se il documento da comporre contiene molte pagine, molti capitoli, eccetera. Conviene predisporre un 'master file' che contenga solo il preambolo e la lista dei file componenti.

```
\documentclass[⟨opzioni⟩]{⟨classe⟩}
\usepackage[⟨opzioni⟩]{⟨pacchetto⟩}[⟨data⟩]
...
\includeonly{⟨lista di file⟩}
\begin{document}
\include{⟨primo file⟩}
\include{⟨secondo file⟩}
...
\end{document}
```

La ⟨*lista di file*⟩ è un elenco di nomi di file separati da virgole. La sequenza di comandi `\include`, ognuno con il suo nome di file, esegue la compilazione dei file, tranne di quelli che *non* sono elencati nella ⟨*lista di file*⟩. Al limite si può compilare un file alla volta; i file già compilati, anche se non vengono più elaborati, hanno lasciato le loro tracce nei rispettivi file `.aux`, quindi, a meno che non siano intervenute modifiche, LaTeX riesce a risolvere tutti i riferimenti incrociati e a mantenere la coerenza dei numeri delle pagine e di tutti i contatori specificati con i comandi specifici di LaTeX.

È chiaro che, prima di terminare la lavorazione di un documento, è importante eseguire la compilazione di tutto il documento, cosicché si è sicuri che tutti i riferimenti e tutti i valori dei contatori e delle pagine siano corretti; questo lavoro, va fatto comunque per predisporre alla fine della lavorazione la compilazione dell'indice analitico (a meno che non si ricorra ai pacchetti *imakeidx* oppure *indextools*, che consentono di comporre uno o più indici analitici in contemporanea alla compilazione del documento).

Ci si ricordi che ogni comando `\include` per prima cosa esegue un comando `\clearpage` che svuota le code di oggetti flottanti e comincia la compilazione del nuovo file su una pagina nuova. Se *non* è questo quello che si vuole ottenere, si faccia uso del comando

```
\input{⟨file⟩}
```

che non consente la compilazione selettiva, ma consente di proseguire la compilazione del testo dal punto preciso nel quale si trovava LaTeX nel momento in cui ha cominciato ad eseguire il comando `\input`.

Per risolvere le dipendenze di un dato documento da certi file di macro, è disponibile l'unico ambiente che può precedere il comando `\documentclass`

```
\begin{filecontents}[⟨opzioni⟩]{⟨nome file⟩}
⟨corpo del file⟩
\end{filecontents}
```

Con le ⟨*opzioni*⟩, in particolare con *noheader* e *overwrite* si controlla che durante l'esecuzione il file ⟨*nome file*⟩ venga o non venga corredato di un 'header' costituito da una riga commentata con il solito segno %; e che sia o non sia riscritto il file se esiste già nella cartella del file principale.

Il comando `\listfiles` serve per elencare a schermo e nel file `.log` tutti i nomi dei file che vengono via via aperti (tranne quelli di servizio) in modo che si possa seguire e controllare che l'esecuzione della compilazione proceda regolarmente.

29.13 Indice analitico e glossario

Come descritto nel capitolo 12 per comporre uno o più indici analitici è opportuno servirsi del programma makeindex; nello stesso capitolo viene anche descritto come servirsi di quel programma per compilare un glossario. Tuttavia qui si richiamano i comandi necessari.

29.13.1 Indice analitico

La composizione dell'indice analitico avviene mediante l'ambiente *theindex*

```
\begin{theindex}
\input{⟨indexfile⟩.ind}
\end{theindex}
```

Più comodamente si usa il pacchetto *makeidx* e si ordina la composizione dell'indice analitico mediante il comando `\printindex`. Ancora più comodamente si usa uno dei pacchetti *imakeidx* o *indextools* che consentono di sfruttare la proprietà dei sistemi TeX moderni di eseguire dei comandi di sistema così da poter generare uno o più indici analitici con una sola esecuzione del programma di composizione.

Però per raccogliere le voci da indicizzare bisogna usare i seguenti comandi:

`\makeindex` inserito nel preambolo ordina di eseguire effettivamente la raccolta delle voci attraverso il comando `\index`, che altrimenti sarebbe completamente inerte.

`\index` serve per raccogliere le voci da indicizzare con la sintassi

```
\index{⟨voce⟩}
```

Il comando `\index` deve essere scritto senza spazi interposti alla fine della parola di testo che si vuole indicizzare. La ⟨*voce*⟩ non contiene solo la parola da indicizzare, ma anche le ulteriori informazioni di cui necessita il programma makeindex per formattare convenientemente la voce come voce principale, oppure secondaria, oppure di terzo livello, per scegliere il font con cui scrivere la voce, per scegliere la chiave di indicizzazione, per segnalare, nel caso di voci secondarie o di terzo livello, sotto quale voce primarie esse debbano essere elencate, per dare uno stile al numero di pagina, eccetera; si veda a questo proposito la documentazione del programma makeindex dando da terminale il comando `texdoc makeindex`..
Chiaramente se si usano i pacchetti *imakeidx* o *indextools* per produrre diversi indici analitici bisogna specificare con un parametro facoltativo in quale indice si deve porre la voce particolare marcata con `\index`; per questo è meglio consultare la documentazione di quei pacchetti.

29.13.2 Glossario

Per comporre un glossario bisogna procedere in modo simile a quello che si usa per l'indice analitico; i comandi che permettono di raccogliere le voci sono

`\makeglossary` inserito nel preambolo consente l'effettiva raccolta delle voci mediante l'attivazione del comando `\glossary`, che altrimenti sarebbe totalmente inattivo.

`\glossary` serve per raccogliere le voci da inserire nel glossario, con la sintassi:

```
\glossary{⟨voce⟩}
```

Evidentemente anche per i glossari esistono appositi pacchetti, come, per esempio, *glossaries* che consentono di lavorare con questi particolari oggetti in modo molto sofisticato.

29.14 Compilazione interattiva

LaTeX consente un certo livello di interattività durante la compilazione; a parte gli errori, che chiedono al compositore di intervenire, LaTeX consente anche di emettere dei messaggi sul terminale e di leggere le risposte introdotte dal compositore.

`\typeout` con la sintassi

```
\typeout{⟨messaggio⟩}
```

consente di scrivere un messaggio sullo schermo del terminale; questo messaggio può avere gli scopi più diversi, ma non consente nessuna interattività; piuttosto può preparare un dialogo interattivo.

`\typein` con la sintassi

```
\typein[⟨comando⟩]{⟨messaggio⟩}
```

emette il ⟨*messaggio*⟩ sullo schermo; se è specificato anche il ⟨*comando*⟩ il programma si arresta e attende che l'operatore scriva qualche cosa sulla tastiera (presumibilmente quanto il ⟨*messaggio*⟩ suggerisce di scrivere fra una rosa di possibili scelte); non appena il compositore preme il tasto Invio, il testo introdotto viene assegnato al ⟨*comando*⟩, esattamente come se il tutto eseguisse le seguenti operazioni

```
\typeout{⟨messaggio⟩}
\newcommand{⟨comando⟩}{⟨testo introdotto⟩}
```

Il ⟨*comando*⟩ deve essere una valida sequenza di controllo, cioè una stringa esclusivamente letterale preceduta dal backslash, oppure un solo segno non letterale preceduto dal backslash, oppure un carattere attivo.

Sulla base di ciò che è stato assegnato al ⟨*comando*⟩ il programma può svolgere diverse azioni, che devono essere programmate con determinate intenzioni da chi ha scritto quel `\typein`.

29.15 Interruzione di riga e di pagina

29.15.1 Interruzione di riga

I comandi

```
\linebreak[⟨numero⟩]
\nolinebreak[⟨numero⟩]
```

consentono o vietano, rispettivamente, di interrompere una riga. Se si specifica il ⟨*numero*⟩ la loro azione è rinforzata o indebolita a seconda del valore del numero stesso. Il numero 0 rappresenta un debole incoraggiamento ad operare, mentre il numero 4 rappresenta l'indicazione di eseguire incondizionatamente l'azione specificata; l'assenza del numero equivale a specificare il numero 4. Attenzione: `\linebreak` interrompe la linea ma lascia che il programma cerchi di giustificarla, per cui se la gomma interparola deve essere allargata troppo, viene emesso un messaggio di `Underfull \hbox`.

Una riga può venire interrotta anche con i comandi

```
\\[⟨spazio⟩]
\\*[⟨spazio⟩]
\newline
```

Essi permettono di interrompere una riga per andare decisamente a capo. Non viene eseguito nessun tentativo di giustificare la riga, nemmeno se manca poco al margine destro. La forma asteriscata impedisce un fine pagina subito dopo.

Per tutti questi comandi la possibilità di eseguire la cesura delle parole in fin di riga è essenziale. In certe circostanze non basta consentire la cesura, ma bisogna aumentare le tolleranze di composizione, cioè bisogna accontentarsi di una maggiore bruttezza di una o più righe. Ciò si ottiene con i comandi

```
\sloppy
\fussy
```

Il primo aumenta le tolleranze di composizione a valori enormi, che per LaTeX equivalgono ad infinito. Il secondo ripristina i valori normali.

29.15.2 Interruzione di pagina

I comandi

```
\pagebreak[⟨numero⟩]
\nopagebreak[⟨numero⟩]
```

Agiscono come `\linebreak` ma riferendosi alla pagina, non alla riga. Il ⟨*numero*⟩ ha lo stesso significato con 0 che indica un blando incoraggiamento ad eseguire, e 4 che ordina di eseguire incondizionatamente il comando.

Per l'interruzione di pagina non si può agire sulla cesura, ma si può temporaneamente allungare la gliglia di stampa rispetto al normale:

```
\enlargethispage{⟨lunghezza⟩}
\enlargethispage*{⟨lunghezza⟩}
```

Il comando senza asterisco allunga semplicemente la griglia; per ovvi motivi ⟨*lunghezza*⟩ dovrebbe essere un multiplo intero di `\baselineskip`, cioè dello scartamento normale con il font di default. Tuttavia il comando asteriscato cerca di allungare la gliglia di stampa di quanto specificato, ma cerca anche di stringere il più possibile gli spazi bianchi verticali al fine di non allungare la griglia, ovvero di allungarla il meno possibile. Specialmente componendo sul verso e sul recto è importante che le gliglie siano della stessa altezza; se i margini laterali sono consistenti, una riga in più in una delle due pagine si nota appena, tuttavia sarebbe meglio evitarlo. Il comando asteriscato potrebbe essere la soluzione se la pagina nella quale esso compare ha abbastanza spazi bianchi.

I comandi

```
\newpage
\clearpage
\cleardoublepage
```

chiudono la pagina corrente; gli ultimi due scaricano anche le eventuali code di oggetti flottanti ancora non svuotate; il comando `\cleardoublepage` eventualmente emette anche una pagina bianca al fine di consentire di iniziare la pagina successiva sul recto. `\cleardoublepage` è tipicamente il primo comando che viene eseguito implicitamente quando si esegue il comando di inizio di un nuovo capitolo e anche quando si immette un file con `\include`.

Sono disponibili anche dei comandi presi a prestito da Plain TeX, ma ancora presenti nel mark up LaTeX:

`\smallbreak` esegue un salto pagina solo se prima era stato introdotto nel testo uno spazio verticale pari a `\smallskipamount`.

`\medbreak` esegue un salto pagina solo se prima era stato introdotto nel testo uno spazio verticale pari a `\medskipamount`.

`\bigbreak` esegue un salto pagina solo se prima era stato introdotto nel testo uno spazio verticale pari a `\bigskipamount`.

`\goodbreak` indica a LaTeX che questo è un buon punto per spezzare la pagina; LaTeX effettivamente esegue il salto se manca poco alla fine naturale della pagina, se, cioè non resta una pagina pesantemente mozza, ma le mancano pochissime righe per completare la giustezza verticale; questo comando agisce inserendo una penalità negativa che quindi incoraggia l'esecuzione di un fine-pagina; talvolta questo comando, contrariamente alle aspettative tronca la pagina a metà o anche prima. Chi scrive preferisce usare il comando `\goodpagebreak` che agisce, invece, calcolando quanto manca per completare la giustezza ed eseguendo un comando `\newpage` solo se quanto manca è inferiore ad un (piccolo) numero specificato di righe; il suo difetto è che deve agire in modo verticale; tuttavia non sarebbe difficile introdurre una variante al comando `\goodpagebreak` descritto in precedenza, per poter inserire un salto pagina incondizionato dove lo si inserisce, anche a metà di un capoverso.

`\filbreak` sempre se manca poco alla fine della pagina, questo comando indica a LaTeX che può eseguire un salto pagina inserendo anche un po' di gomma verticale in modo che non vengano emessi messaggi di `Underfull vbox`; nello stesso tempo, se il punto non è buono per interrompere la pagina `\filbreak` non fa nulla. Purtroppo raramente ci si può affidare a questo comando, ma `\goodpagebreak` fa le stesse cose senza sbagliarsi.

Si consiglia di usare questi comandi solo alla fine della lavorazione di un documento per correggere a mano alcune piccole cose che il programma da solo non è capace di gestire. I casi in cui il compositore è costretto a ricorrere a questi comandi sono rarissimi in testi 'normali'; possono essere frequenti in testi che contengono molta matematica. Tuttavia `\enlargethispage` con o senza asterisco consente di rimediare ad un capitolo la cui ultima riga o ultime due

righe cadono nell'ultima pagina, peggio ancora se questa pagina è dispari. Se la pagina è pari questa eventualità non è tanto bella ma può essere accettata; se esse cadono su una pagina dispari, questa resta quasi bianca, e il capitolo successivo inizia in generale di nuovo su una pagina dispari, con il risultato che appaiono due pagine bianche o quasi bianche di seguito; decisamente da evitare.

Piuttosto se i capoversi finali del capitolo problematico sono sufficientemente lunghi si può convincere il programma a comporli su una o due righe di più o di meno, senza forzare allargamenti o accorciamenti improponibili degli spazi interparola, specificando il comando primitivo `\looseness` seguito da un (piccolo) numero positivo o negativo; per esempio scrivendo `\looseness 1` dentro o in assoluta adiacenza ad un capoverso, il programma cercherà di allungare quel solo capoverso di una riga purché il farlo non aumenti la 'bruttezza' del capoverso oltre alla tolleranza valida in quel momento. Il valore di `\looseness` viene rimesso a zero automaticamente alla fine del capoverso, quindi vale solo per quel capoverso per il quale è stato specificato. Per esempio, questo capoverso è stato composto con un valore unitario di questo parametro.

29.16 Scatole

Una *scatola* è un oggetto che può contenere diversi segni, anche un testo formato da diverse righe, ma questo testo, una volta inscatolato, viene trattato dal programma come se fosse un oggetto unico e non lo spezzerà mai alla fine di una riga o di una pagina.

29.16.1 Scatole di uso immediato

I comandi per mettere del ⟨*testo*⟩ dentro una scatola sono

```
\mbox{⟨testo⟩}
\makebox[⟨larghezza⟩][⟨posizione⟩]{⟨testo⟩}
\fbox{⟨testo⟩}
\framebox[⟨larghezza⟩][⟨posizione⟩]{⟨testo⟩}
\begin{lrbox}{⟨scatola⟩}⟨testo⟩\end{lrbox}
```

I comandi che cominciano con la lettera 'f' inseriscono una cornice attorno al testo; quelli che cominciano con la lettera 'm' non inseriscono nessuna cornice. Se la ⟨*larghezza*⟩, che è facoltativa, viene specificata, il testo viene inserito dentro una scatola della larghezza specificata, altrimenti i comandi che accettano questo argomento facoltativo si comportano come quelli che non lo richiedono. Se viene specificata la ⟨*larghezza*⟩ allora si può specificare anche la posizione mediante una sola delle lettere seguenti: `l` per collocare il testo a sinistra, `r` per collocare il testo a destra, `s` per distribuire il testo, allargando o restringendo lo spazio interparola, lungo tutta la ⟨*larghezza*⟩; se non viene specificata nessuna posizione il testo risulta centrato dentro la ⟨*larghezza*⟩. Si confrontino i casi seguenti.

`\framebox[50mm][l]{testo a sinistra}`	testo a sinistra
`\framebox[50mm][r]{testo a destra}`	testo a destra
`\framebox[50mm]{testo centrato}`	testo centrato
`\framebox[50mm][s]{testo spaziato}`	testo spaziato

Lo spessore della linea che forma la cornice delle scatole è conservata nel parametro dimensionale `\fboxrule` e la distanza della cornice dal testo è conservata dentro il parametro dimensionale `\fboxsep`. Attenzione: lo spessore di 1 pt per la cornice dà luogo a una cornice abbastanza scura: testo . Quindi non è il caso di specificare spessori maggiori se non a ragion veduta.

Si può sostituire l'indicazione esplicita della⟨*larghezza*⟩ nelle scatole che l'accettano mediante quattro nuovi comandi che si riferiscono tutti alle dimensioni della scatola composta con `\mbox` che contiene lo stesso ⟨*testo*⟩; esse sono:

`\height` ovvero l'altezza della scatola composta con `\mbox{`⟨*testo*⟩`}`
`\depth` ovvero la profondità di quella scatola
`\totalheight` ovvero la somma di `\height` e di `\depth`
`\width` ovvero la larghezza di quella scatola.

29.16.2 Scatole per conservare del testo

Oltre ad assegnare nomi simbolici alle scatole mediante il comando `\newsavebox`, LaTeX consente di memorizzare alcune scatole in appositi registri, e di usarne il contenuto in diversi modi; chiaramente le macro del mark-up di LaTeX sono definite in termini di comandi primitivi del sottostante linguaggio TeX, ma mentre questi comandi permettono di fare cose raramente necessarie per l'utente (il quale se sente la necessità di usarle è un utente abbastanza avanzato per essersi documentato sul TeXbook), le macro di LaTeX servono per eseguire le operazioni più frequenti; esse sono le seguenti:

`\newsavebox` con la sintassi

`\newsavebox{`⟨*scatola*⟩`}`

alloca un registro-scatola ed assegna il suo indirizzo alla ⟨*scatola*⟩; questa ⟨*scatola*⟩ è il nome di un registro"=scatola formalmente simile e conforme nella sua forma a tutte i nomi di macro di LaTeX quindi deve cominciare con una barra rovescia e proseguire con una stringa di sole lettere maiuscole o minuscole; è evidente che è bene che questa stringa abbia un significato mnemonico, cosicché l'utente sa dal nome a che scopo ha definito quella scatola.

`\sbox` con la sintassi

`\sbox{`⟨*scatola*⟩`}{`⟨*testo*⟩`}`

carica la ⟨*scatola*⟩ con il ⟨*testo*⟩; si noti che qui ⟨*scatola*⟩ può essere tanto un nome assegnato ad un registro-scatola mediante `\newsavebox` (e questo varrà anche per i comandi successivi), ma potrebbe anche essere direttamente un indirizzo assoluto di un registro-scatola. Questa seconda possibilità è appunto una possibilità, ma è fortemente sconsigliabile, perché specificando un indirizzo "sbagliato" si rischia di mettere qualcosa in un registro-scatola usato dal nucleo di LaTeX, cosa che potrebbe produrre errori difficilmente diagnosticabili.

Attenzione: va detto qui, ma lo si potrebbe ripetere anche per i prossimi comandi: il processo di allocazione dei registri-scatola mediante `\newsavebox` permette di associare un nome ⟨*scatolla*⟩ con gli indirizzi fino al 254; non permette di usare il registro 255 che è riservato per il nucleo di LaTeX; permette di allocare altri registri scatola purché non superino il numero 32 767. Si sconsiglia fortemente di usare registri scatola chiamandoli per numero.

Si noti ancora che `\sbox` funziona in modo simile a `\mbox` solo che quest'ultimo non carica la scatola in memoria ma la usa subito per il testo che si sta componendo.

`\savebox` continua la similitudine di questo comando con quello di `\makebox`; esso ha la sintassi:

```
\savebox{⟨scatola⟩}[⟨larghezza⟩][⟨posizione⟩]{⟨testo⟩}
```

ed agisce come `\makebox`, a parte la memorizzazione del contenuto in un registro-scatola; i suoi argomenti obbligatori e facoltativi hanno gli stessi significati.

`\usebox` con la sintassi:

```
\usebox{⟨scatola⟩}
```

permette di usare il contenuto della ⟨*scatola*⟩ ma questo contenuto rimane memorizzato nel suo registro ed è usabile con successivi comandi `\usebox`; tuttavia se il contenuto della ⟨*scatola*⟩ è stato assegnato mentre si era dentro un gruppo o dentro un ambiente, come si esce dal gruppo o dall'ambiente il contenuto del registro viene ripristinato al valore che esso aveva prima di aprire il gruppo o l'ambiente.

29.16.3 Ambienti e comandi per scatole particolari

L'ambiente *lrbox* è un po' particolare; ha la sintassi seguente:

```
\begin{lrbox}{⟨scatola⟩}
⟨testo⟩
\end{lrbox}
```

e serve per comporre il ⟨*testo*⟩ dentro la ⟨*scatola*⟩ specificata con il suo numero o il suo nome simbolico precedentemente dichiarato mediante il comando `\newsavebox`; in un certo senso esso si comporta come se venissero eseguiti i comandi primitivi:

`\setbox{`⟨*scatola*⟩`}\hbox{`⟨*testo*⟩`}`

ma in realtà fa molto di più: infatti ⟨*testo*⟩ può essere qualunque cosa che abbia senso in modo orizzontale, compreso, per esempio, un ambiente *minipage* (vedi poco più avanti); il fatto che non sia necessario usare le graffe per delimitare il ⟨*testo*⟩, rende questo ambiente particolarmente utile per definire ambienti che ricorrono a comandi i cui argomenti debbano essere delimitati da graffe vere, non sostituibili con graffe logiche. Per esempio se si volesse comporre un testo di diverse righe composto dentro una cornice, non lo si potrebbe fare direttamente con il comando `\fbox` ma lo si potrebbe fare definendo un ambiente *riquadro* in questo modo:

```
\newenvironment{riquadro}[1][\linewidth]{% apertura
\begin{lrbox}{0}\begin{minipage}{\dimexpr#1-2\fboxsep-2\fboxrule}
}{% chiusura
\end{minipage}\end{lrbox}\fbox{\usebox{0}}\relax
}
```

e questo nuovo ambiente verrebbe usato così:

`\begin{riquadro}[`⟨*larghezza*⟩`]`
⟨*testo*⟩
`\end{riquadro}`

Per esempio:

Questo è un testo riquadrato mediante l'ambiente *riquadro* definito poco sopra.

Il procedimento si basa sul fatto che l'ambiente *lrbox* carica una scatola con una "paginetta" di giustezza specificata (in mancanza della specificazione dell'argomento facoltativo viene usata la giustezza corrente del testo, `\linewidth`) composta con *minipage* (vedi poco oltre), poi, dopo aver finito questo caricamento, usa la scatola come argomento di `\fbox` che le disegna la cornice attorno.

Per inserire del testo dentro scatole che contengono materiale 'verticale' (in pratica una successione verticale di righe, uno o più capoversi) si può usare il comando `\parbox` o l'ambiente *minipage*:

`\parbox[`⟨*allineamento*⟩`][`⟨*altezza*⟩`][`⟨*pos.interna*⟩`]{`⟨*giustezza*⟩`}{`⟨*testo*⟩`}`

oppure

`\begin{minipage}[`⟨*allineamento*⟩`][`⟨*altezza*⟩`][`⟨*pos.interna*⟩`]{`⟨*giustezza*⟩`}`
⟨*testo*⟩
`\end{minipage}`

dove il parametro ⟨*allineamento*⟩ indica l'allineamento della scatola con il testo circostante; `t` indica che la riga di base della prima riga della scatola è allineata con la riga di base del testo circostante; `b` indica che la linea di base dell'ultima riga della scatola è allineata con la riga di base del testo circostante; con il codice `c` predefinito, e quindi senza specificare nulla, l'asse matematico della scatola è allineato con l'asse matematico del testo circostante. Facoltativamente si può specificare l'⟨*altezza*⟩ della scatola; se questa altezza viene specificata, allora si può usare il terzo parametro facoltativo ⟨*pos.interna*⟩; le lettere di posizionamento sono le stesse di quelle usate per l'allineamento dell'intera scatola, ma si riferiscono a dove il testo contenuto nella scatola stessa viene collocato verticalmente rispetto ai suoi bordi ideali. Va infine specificata la ⟨*giustezza*⟩ della scatola, in pratica la sua larghezza, e il ⟨*testo*⟩ che essa deve contenere.

La differenza sostanziale fra il comando e l'ambiente è la seguente: il comando `\parbox` non comincia a comporre il ⟨*testo*⟩ al suo interno, finché non l'ha letto tutto. Tutto ciò non avviene con l'ambiente *minipage* il quale gestisce il ⟨*testo*⟩ in modo differente. In compenso *minipage* in quanto ambiente, richiede una 'amministrazione' un poco più complessa che richiede un maggior tempo di elaborazione. Parlando di microsecondi, non è il caso di formalizzarsi troppo, visti i numerosissimi vantaggi che offre l'ambiente rispetto al comando. L'ambiente *minipage* può anche contenere delle note interne alla 'paginetta' che esso compone; queste note vengono composte con i soliti comandi; solo che la numerazione delle note avviene con lettere corsive in posizione di apice, in modo da distinguerle bene dalle note di piè di pagina, generalmente composte con esponenti numerici o con simboli non letterali.

Si noti che se il contenuto di una *minipage* comincia con una equazione in display, essa conserva lo spazio che precede l'equazione; se non si vuole questo spazio si cominci la *minipage* con `\vspace{-\abovedisplayskip}`.

Il comando `\rule` compone una scatola particolare: essa, infatti, è completamente nera. La sintassi è:

`\rule[`⟨*rialzo*⟩`]{`⟨*base*⟩`}{`⟨*altezza*⟩`}`

La ⟨*base*⟩ e l'⟨*altezza*⟩ sono la base e l'altezza del rettangolo nero; se una delle due è nulla il rettangolo è invisibile, ma l'oggetto continua a mantenere l'altra dimensione; se la base è nulla si ottiene uno *strut*, che in gergo tipografico italiano si chiama *pilastrino*. In questo testo si è mantenuto il nome inglese per ricordare più facilmente i nomi dei comandi che inseriscono *strut* vari sia nelle tabelle, sia in matematica. Il rettangolo nero (o invisibile) risulta rialzato di ⟨*rialzo*⟩ se questa lunghezza è positiva, o ribassato se essa è negativa.

Infine un comando serve per collocare del ⟨*testo*⟩ in una scatola che viene alzata o abbassata a piacere e le si possono assegnare dimensioni a piacere, indipendentemente dal testo che la scatola contiene.

`\raisebox{`⟨*rialzo*⟩`}[`⟨*altezza*⟩`][`⟨*profondità*⟩`]{`⟨*testo*⟩`}`

La ⟨*profondità*⟩ può venire specificata solo se si specifica anche l'⟨*altezza*⟩.

29.16.4 Cenno ai comandi per le scatole con il linguaggio nativo di TeX

Come accennato sopra, il linguaggio nativo di TeX, su cui si basano le macro del mark-up di LaTeX, sono più numerose e svolgono anche altri compiti che con i comandi di LaTeX non si possono svolgere. La descrizione che segue è molto sommaria; per approfondire bisogna consultare il TeXbook.

\hbox è simile a **\mbox** ma se viene usato in modo verticale il modo di composizione non viene modificato, mentre con **\mbox** si passa al modo orizzontale. L'argomento di **\hbox** può anche venire racchiuso fra graffe simboliche, per esempio **\bgroup** e **\egroup**, cosicché nella definizione di ambienti la graffa simbolica di apertura può stare fra i comandi di apertura e la graffa simbolica di chiusura può stare fra i comandi di chiusura, senza che siano strettamente appaiate all'interno dei due argomenti di definizione del comando **\newenvironment**. La sua sintassi è sostanzialmente uguale a quella di **\mbox**, ma in linguaggio TeX è ammessa una variante

```
\hbox to⟨larghezza⟩{⟨testo⟩}
```

che si comporta come **\makebox**; la posizione del testo dentro la scatola viene definito specificando a sinistra e a destra del ⟨*testo*⟩ della "gomma" sufficiente per rispettare la ⟨*larghezza*⟩ specificata; normalmente questa "gomma" viene specificata con **\hss** il cui nome significa *horizontal stretch and shrink*; si tratta di una lunghezza elastica di lunghezza naturale nulla ma con allungamento e accorciamento illimitati.

\vbox compone una scatola verticale come farebbe **\parbox[b]**, ma il suo argomento può essere delimitato da graffe simboliche.

\vtop compone una scatola verticale come farebbe **\parbox[t]**, ma il suo argomento può essere delimitato da graffe simboliche.

\vcenter compone una scatola verticale come farebbe **\parbox[c]** o quando non è specificata nessuna opzione di posizionamento, ma il suo argomento può essere delimitato da graffe simboliche. Può essere usato solo in modo matematico, visto che il suo allineamento verticale è basato sull'asse matematico della scatola che viene allineato con l'asse matematico del testo circostante.

Tutti e tre questi comandi ammettono le varianti **\vbox to**. **\vtop to** e **\vcenter to**, ma la "gomma" deve essere ora specificata con **\vss** (*vertical stretch and shrink*) per collocare il ⟨*testo*⟩ in alto o in basso o al centro della lunghezza (altezza) specificata.

\newbox del tutto simile a **\newsavebox**.

\setbox è simile a **\sbox** per le scatole orizzontali di LaTeX, ma in TeX serve anche per le scatole verticali; la sintassi è:

```
\setbox⟨scatola⟩⟨comando scatola⟩{⟨testo⟩}
```

dove ⟨*scatola*⟩ rappresenta l'indirizzo assoluto o il nome di un registro-scatola allocato mediante `\newbox`; il ⟨*comando scatola*⟩ è uno dei comandi TEX per inscatolare il testo: `\hbox`, `\vbox`, `\vtop`, `\vcenter` (questo con qualche cautela a causa dell'ambiente matematico necessario per usarlo).

`\box` serve per inserire la ⟨*scatola*⟩ identificata dal suo argomento nel testo che si sta componendo; il corrispondente registro-scatola viene svuotato. La sintassi per questo e i comandi successivi è

```
\box⟨scatola⟩
```

dove ⟨*scatola*⟩ è l'indirizzo assoluto di un registro-scatola oppure il suo nome simbolico.

`\unhbox` serve per "disinscatolare" il ⟨*testo*⟩ contenuto nella ⟨*scatola*⟩; dopo aver usato questo comando il registro-scatola è vuoto; la differenza rispetto a `\box` è che quest'ultimo non estrae il contenuto dalla scatola, ma lascia il ⟨*testo*⟩ inscatolato e gestibile come un unico oggetto; con `\unhbox` i singoli token contenuti nella scatola orizzontale vengono estratti e possono venire ricomposti a piacere. Un esempio tipico di questa differenza è contenuto nella definizione standard della didascalia con le classi standard; in questo caso, infatti, l'argomento obbligatorio di `\caption` che corrisponde al testo (lungo) della didascalia viene memorizzato in una scatola; viene misurata la larghezza di tale scatola e se è inferiore alla giustezza del testo corrente, la scatola viene estratta in una riga centrata; altrimenti il contenuto della scatola viene estratto e ricomposto in una scatola verticale `\vtop` (per allineare la prima riga della didascalia con il suo titolino) con la giustezza specifica per le didascalie e poi la scatola verticale viene copiata dove va composta la didascalia. Ovviamente l'utente non si accorge di tutto questo lavorio dietro le quinte, e questo è uno dei pregi del sistema di macro che costituiscono il make-up di LATEX.

`\unvbox` serve per "disinscatolare" le scatole orizzontali (le righe) che costituiscono il contenuto di una scatola verticale; quindi non si estrae il testo immesso nella scatola, ma si estraggono le righe nelle quali è stato composto il testo. Alla fine dell'operazione la scatola è vuota.

`\unhcopy` agisce come `\unhbox` ma non svuota il registro; si comporta in modo simile al comando di LATEX `\usebox`.

`\unvcopy` agisce come `\unvbox` ma non svuota il registro.

`\vsplit` serve per estrarre una parte del contenuto di una scatola verticale estraendola dal registro originale; si usa sostanzialmente così:

```
\setbox⟨scatola2⟩ = \vsplit ⟨scatola1⟩ to ⟨dimensione⟩
```

dove ⟨*scatola1*⟩ è una scatola verticale che contiene un certo numero di righe (scatole orizzontali) di altezze variabili se sono state composte con font di corpi diversi; ⟨*dimensione*⟩ non è un numero, ma è l'altezza complessiva delle righe che si vogliono estrarre da ⟨*scatola1*⟩ per metterle

in ⟨*scatola2*⟩. Tipicamente questa è l'operazione eseguita dalla routine di uscita di LaTeX quando deve estrarre una pagina composta da spedire al file di uscita prelevando la parte superiore della scatola-registro 255, riservato da LaTeX per accumulare via via i capoversi del testo immesso dall'utente. Ovviamente la routine di uscita non si limita ad estrarre il testo dalla scatola 255, per comporre la pagina in uscita, visto che deve inserire gli oggetti mobili, le note, le testatine e i piedini, insomma, visto che deve confezionare la pagina completa di tutto quello che occorre, non solo del testo.

29.16.5 Operazioni di misura sulle scatole

Le scatole possono servire a molti scopi; esse sono un ingrediente essenziale del modo di funzionare dei programmi di composizione del sistema TeX, ma esse possono venire usate dall'utente in vari modi.

Per esempio, esse possono servire per prendere le misure dell'ingombro di certi oggetti da inserire nel documento; basta metterli in una scatola e chiedere al programma di fornirne le misure; i comandi di LaTeX che si sono già descritti sono `\settowidth`, `\settoheight` e `\settodepth`, i cui argomenti sono un nome simbolico di una registro-dimensione e un testo o comunque un oggetto da misurare. Per non usare inutilmente registri interni del programma, si può fare anche in un altro modo, se si deve poi usare l'oggetto in un secondo tempo; basta memorizzare l'oggetto in una scatola, poi si usano i comandi di TeX`\ht`, `\wd` e `\dp` per misurare rispettivamente l'altezza, la larghezza e la profondità; sono sostanzialmente le stesse operazioni che svolgono i comandi di LaTeX, ma ognuno ogni volta prende il testo e lo memorizza in una scatola; si esamini questo esempio, dove vogliamo misurare le dimensioni del logo GuIT nel formato `\Huge`, ma vogliamo poter utilizzare questo grande logo più volte senza doverlo ricomporre:

```
% nel preambolo
\newsavebox{\scatolaGuIT}
\savebox{\scatolaGuIT}{\Huge \GuIT}}

% Nel corpo del documento
\dots\
Il logo di \GuIT\ in formato \texttt{\textbackslash Huge},
\usebox{\scatolaGuIT}, ha le dimensioni seguenti:
altezza \the\ht\scatolaGuIT; larghezza \the\wd\scatolaGuIT;
profondità \the\dp\scatolaGuIT.
\dots
```

che permette di ottenere:

> ...
> Il logo di GuIT in formato `\Huge`, GUIT, ha la dimensioni seguenti:

altezza 17.00131pt; larghezza 41.97949pt; profondità 7.64856pt.
...

Il commando `\the` è già stato descritto nella pagina 744.

Esistono altri comandi nel linguaggio TEX per fare cose ancora più insolite con le scatole, ma si ritiene che quanto esposto sia sufficiente per dare una idea di che cosa il linguaggio TEX sia capace. Il lettore interessato deve documentarsi convenientemente sul TEXbook, che è la fonte primaria di informazione sul linguaggio TEX.

29.17 Disegni e colori

29.17.1 Disegni

Come già specificato l'ambiente *picture* nativo di LATEX non è in grado di fare altro che semplici disegni, con l'inclinazione dei segmenti e dei vettori molto limitata dagli speciali font che servono per disegnarli; anche i cerchi sia pieni sia vuoti sono un problema, se non nelle situazioni più semplici; i rettangoli ad angoli arrotondati soffrono delle stesse limitazioni dei cerchi; le linee arbitrarie possono essere eseguite solo con le curve di Bézier quadratiche, quindi con certe limitazioni sulle quali qui non è il caso di insistere. Dal 2003 è disponibile il pacchetto *pict2e*, annunciato nel 1994 dallo stesso Leslie Lamport, il creatore di LATEX, che toglie tutte queste limitazioni; esso ha assunto una conformazione stabile dal 2004; quindi si raccomanda caldamente di usare quel pacchetto se si desiderano eseguire semplici disegni. Nel 2009 il pacchetto *pict2e* è stato ulteriormente arricchito di altri comandi che permettono ora di comporre disegni piuttosto elaborati; *pict2e* è parte integrante del sistema TEX, ma anche se non ancora incorporato nel nucleo di LATEX, oggi va sempre caricato e usato. Vedi anche il capitolo 30 per le ultimissime novità.

Se si vuole disegnare qualcosa di più elaborato ancora, si ricorra al pacchetto *pgf* e al suo ambiente *tikzpicture*; questo pacchetto consente di disegnare praticamente qualsiasi cosa. Se non dovesse bastare c'è sempre il pacchetto *PSTricks* che è limitato solo dal fatto che il linguaggio PostScript su cui si basa non è onnipotente, ma quasi.

Qui, pertanto, non si darà nessuna ulteriore informazione sull'ambiente *picture*; gli esempi svolti nel testo dovrebbero essere autoesplicativi. Si invita invece il lettore a documentarsi sui pacchetti *pict2e*, *pgf* e sul suo ambiente *tikzpicture*, nonché sul pacchetto *PSTricks*.

29.17.2 Colori e grafica

LATEX è dotato di una serie di pacchetti che consentono di eseguire diverse cose: oltre all'importazione di file grafici esterni, essi mettono a disposizione la gestione del colore e la manipolazione di scatole varie, che normalmente non sono

previste dal mark up LaTeX, ma possono invece venire eseguite direttamente dal programma di composizione. L'intera serie di pacchetti si trova nella cartella `.../tex/latex/graphics/`, mentre la documentazione si trova in `.../doc/latex/graphics/`. Il lettore è invitato a documentarsi sul file `grfguide.pdf`; qui si riportano solo i comandi più importanti.

Si noti che fra questi pacchetti c'è *color* per la gestione del colore; i colori possono essere chiamati per nome o possono essere descritti a livello più basso mediante la specificazione del modello del colore e le componenti dalle quali in questo modello un dato colore è caratterizzato. Il pacchetto *xcolor* estende moltissimo queste funzionalità.

La gestione delle scatole e l'importazione di figure esterne è gestita da altri pacchetti: di questi si è citato qui sempre e soltanto *graphicx*, che è il più comodo da usare, in quanto esso dispone di una interfaccia a parole chiave, rispetto al file che contiene le macro vere e proprie di gestione della grafica, *graphics*. Quest'ultimo può essere usato da solo, ma la sintassi diventa spesso abbastanza complessa per fargli eseguire le cose che sono così semplici con *graphicx*.

`\scalebox{`⟨*scala-orizz.*⟩`}[`⟨*scala-vert.*⟩`]{`⟨*testo*⟩`}` serve per inserire ⟨*testo*⟩ dentro una scatola per poi scalarla del fattore di scala orizzontale; se il fattore di scala verticale non viene specificato l'ingrandimento verticale avviene con lo stesso fattore di scala orizzontale, altrimenti la scatola viene deformata in maniera diversa così da ottenere effetti particolari.

`\resizebox{`⟨*larghezza*⟩`}{`⟨*altezza*⟩`}{`⟨*testo*⟩`}` esegue in pratica la stessa operazione di `\scalebox` solo che invece di specificare dei fattori di scala, vengono specificate le dimensioni finali effettive. Il comando asteriscato agisce in modo da scalare l'altezza totale (altezza più profondità) al valore specificato. Una delle due dimensioni può essere sostituita da un punto esclamativo; in questo caso l'altra dimensione fissa il fattore di scala complessivo in modo che il rapporto di forma non sia modificato.

`\rotatebox[`⟨*lista di chiavi*⟩`]{`⟨*angolo*⟩`}{`⟨*testo*⟩`}` serve per ruotare la scatola che contiene il ⟨*testo*⟩ di un ⟨*angolo*⟩ specificato (in gradi) in senso antiorario; il ⟨*testo*⟩ può essere qualunque cosa: del testo vero e proprio, una figura, una tabella, ...

Nella ⟨*lista di chiavi*⟩ si può anche specificare mediante *origin* il punto attorno al quale eseguire la rotazione; si possono definire delle coordinate, ma ci sono alcuni punti speciali che possono essere rappresentati mediante codici letterali: `t`, `b`, `r`, `l` e `B` rappresentano le solite iniziali di 'top', 'bottom', 'right', 'left', ma `B` è un codice nuovo rispetto a quelli già incontrati altre volte: esso specifica la linea di 'Base', la 'Baseline', la linea, cioè, che attraversa orizzontalmente la scatola in corrispondenza del punto di riferimento; queste lettere sono da usare a due a due; per esempio: `tl` per indicare l'angolo in alto a sinistra, `Br` per indicare l'intersezione della linea di base con il lato destro della scatola, eccetera. Si osservino le tre diverse posizioni che assume la lettera 'Q' quando la si ruoti di 90° attorno ai tre possibili punti di rota-

zione di sinistra: '`bl`' ↺, '`Bl`' ↺ e '`tl`' ↺. Generalmente per ruotare una scritta in verticale, per esempio per metterla a fianco di un asse verticale in un disegno, è conveniente eseguire la rotazione attorno al punto `Bl`.

`\reflectbox{`⟨*text*⟩`}` gira la scatola che contiene il ⟨*testo*⟩ attorno al suo asse verticale in modo che appaia come riflessa allo specchio: 'oıɟɔɔǝqƨ'.

`\includegraphics[`⟨*lista di chiavi*⟩`]{`⟨*file grafico*⟩`}` serve per inserire una scatola che contiene l'immagine descritta nel ⟨*file grafico*⟩ elaborata secondo le numerose chiavi che si possono specificare, ognuna con il suo valore; se si specifica il *bounding box*, il rettangolo 'circoscritto', la versione asteriscata di questo comando permette di tagliare tutto quanto sporge fuori da detto rettangolo. Per questo scopo talvolta è meglio specificare le coordinate del `viewport` o le dimensioni delle strisce da ritagliare con `trim` e poi specificare la chiave `clip`; in questo modo si può avere un controllo migliore di quanto si sta eseguendo.

`\definecolor{`⟨*nome*⟩`}{`⟨*modello*⟩`}{`⟨*valore*⟩`}` serve per dare un ⟨*nome*⟩ ad un colore specificato mediante il suo ⟨*modello*⟩ e il ⟨*valore*⟩ dei parametri che in quel modello identificano il colore scelto. LaTeX considera come modelli di colore i seguenti:

`gray` è il modello di colore grigio identificato da un solo valore, un numero decimale nell'intervallo $0-1$; 0 vuol dire nero e 1 significa bianco; ogni valore intermedio specifica una certa gradazione di grigio.

`rgb` è il modello di colore degli schermi: rosso, verde, blu. I colori sono additivi e il valore che specifica un dato colore in questo modello è dato da tre numeri decimali separati da virgole, tutti e tre nell'intervallo $0-1$; 0 vuol dire 'assenza' di quella particolare componente di colore, 1 significa che quella componente di colore è totalmente satura. Per esempio $1,0,0$ indica il rosso saturo.

`cmyk` è il modello della quadricromia a stampa con i colori sottrattivi; ogni colore è identificato da quattro componenti corrispondenti ciascuna alle componenti cìano (celeste), magenta (lilla), giallo, nero. Ogni componente è pesata con un numero decimale nell'intervallo $0-1$ e la quaterna di valori decimali costituisce una lista di valori separati da virgole.

Con ogni modello di colore, sono predefiniti i seguenti colori: `white`, `black`, `red`, `green`, `blue`, `yellow`, `cyan`, `magenta`. Il driver specifico a seconda se si passi attraverso il programma `dvips` oppure se si proceda con gli altri programmi di composizione, può definire altri colori. I nomi dei colori predefiniti con il driver `dvips` sono contenuti nel file `dvipsnam.def` in `.../tex/latex/graphics/`.

`\color{`⟨*nome*⟩`}` dichiara che da questo momento in poi si userà il colore con il ⟨*nome*⟩ specificato. La sua azione viene delimitata dai gruppi o dagli

ambienti.

`\textcolor{`⟨*nome*⟩`}{`⟨*testo*⟩`}` serve per colorare il ⟨*testo*⟩ con il colore denominato ⟨*nome*⟩.

`\colorbox{`⟨*sfondo*⟩`}{`⟨*testo*⟩`}` agisce come `\fbox`, ma il ⟨*testo*⟩ viene composto su uno sfondo colorato con il colore di nome ⟨*sfondo*⟩.

`\fcolorbox{`⟨*bordo*⟩`}{`⟨*sfondo*⟩`}{`⟨*testo*⟩`}` agisce come `\colorbox` salvo che la cornice della scatola non è nera ma colorata con il colore di nome ⟨*bordo*⟩. I parametri `\fboxsep` e `\fboxrule` che definiscono lo spazio di separazione fra bordo e ⟨*testo*⟩ e lo spessore del bordo, sono gli stessi che regolano le caratteristiche della scatola incorniciata con `\fbox`.

`\pagecolor{`⟨*nome*⟩`}` questo comando *globale* assegna alle pagine intere uno sfondo con il colore denominato ⟨*nome*⟩ e questo colore permane finché non si dichiara un colore differente (eventualmente `white`).

29.18 Selezione dei caratteri

I comandi standard di LaTeX che hanno a che vedere con la scelta dei font verranno descritti qui di seguito. Per gestire i font in modo più professionale, anzi, per caricare altre collezioni di font e per gestirle al meglio, è necessario documentarsi nella guida `fntguide.pdf`.

29.18.1 Scegliere famiglia, forma e serie

I comandi per scegliere la forma e/o la serie da cui trarre i font per la composizione sono raccolti nella tabella 29.4; nella tabella 29.5 sono invece appaiate le dichiarazioni e i corrispondenti comandi. Con gli ultimi aggiornamenti, vedi il capitolo 30, i comandi per la scelta della serie e della forma sono stati estesi.

Le dichiarazioni non possono essere usate in matematica; i comandi possono essere usati in matematica, e sono anche soggetti alla variazione automatica di corpo che compete agli indici e ai pedici di primo e di secondo livello *se e solo se* viene caricato anche il pacchetto *`amsmath`*, ma i risultati possono essere pessimi se i font matematici e quelli testuali non si accordano; per cui se ne sconsiglia l'uso e si raccomanda invece di usare il corretto comando matematico `\mathrm` per i pedici e gli apici che normalmente contengono solo una parola o una abbreviazione, sempre scritti senza accenti. Si può ricorrere anche al comando `\text` fornito dal pacchetto *`amsmath`* e al corrispondente comando `\intertext` per intercalare del testo ad espressioni matematiche incolonnate. Nell'argomento di questi due comandi, se fosse necessario, si può ricorrere tanto ai comandi quanto alle dichiarazioni. Vale la pena di ricordare che tanto `\text` quanto `\intertext` usano il font testuale, non il tondo matematico o simili altri font matematici. Ci si ricordi quindi della necessità di scegliere i font testuali e quelli matematici in modo che si accordino gli uni con gli altri; di default i font CM, EC, LM, cm-super, ed altre simili collezioni di font, si accordano sia in modo testo sia in modo matematico.

`\mdseries`	Serie media	`\upshape`	Forma diritta
`\bfseries`	**Serie nera**	`\itshape`	*Forma corsiva*
`\rmfamily`	Tondo con grazie	`\slshape`	*Forma inclinata*
`\sffamily`	Lineare senza grazie	`\scshape`	MAIUSCOLETTO
`\ttfamily`	`Monospaziato`	`\normalfont`	Font di default

Tabella 29.4: Dichiarazioni per la scelta di famiglia, serie e forma

`\textmd{`⟨*testo*⟩`}`	`\mdseries`	`\textup{`⟨*testo*⟩`}`	`\upshape`
`\textbf{`⟨*testo*⟩`}`	`\bfseries`	`\textit{`⟨*testo*⟩`}`	`\itshape`
`\textrm{`⟨*testo*⟩`}`	`\rmfamily`	`\textsl{`⟨*testo*⟩`}`	`\slshape`
`\textsf{`⟨*testo*⟩`}`	`\sffamily`	`\textsc{`⟨*testo*⟩`}`	`\scshape`
`\texttt{`⟨*testo*⟩`}`	`\ttfamily`	`\textnormal{`⟨*testo*⟩`}`	`\normalfont`

Tabella 29.5: Corrispondenza fra comandi e dichiarazioni

`\tiny`	`\scriptsize`	`\footnotesize`	`\small`	`\normalsize`
`\large`	`\Large`	`\LARGE`	`\huge`	`\Huge`

Tabella 29.6: Dichiarazioni di corpo

29.18.2 Scegliere il corpo

La dimensione dei caratteri è legata a nomi che fanno riferimento al corpo 'normale', quello, cioè, usato di default. Per i corpi inferiori a 10 pt la successione dei corpi procede di 1 pt in 1 pt; per i corpi superiori si procede con una successione geometrica di ragione 1,2; l'opzione di classe *11pt* corrisponde ad un corpo di 10,95 pt, ma il numero magico 10,95 altro non è che $10 \times \sqrt{1{,}2}$. I comandi per la scelta dei corpi relativi a quello normale sono raccolti nella tabella 29.6. Questi comandi non possono venire usati in matematica.

29.18.3 Corpi testuali e matematici

Si ricorda che fra i comandi disponibili (ma raramente usati) è disponibile il comando per associare i quattro corpi che servono per il testo e gli indici superiori e inferiori di primo e di secondo ordine per la matematica:

`\DeclareMathSizes{`⟨*testo*⟩`}{`⟨*textsize*⟩`}{`⟨*subscriptsize*⟩`}{`⟨*subsubscriptsize*⟩`}`

dove al corpo testuale ⟨*testo*⟩ vengono associati i tre corpi per la matematica: ⟨*textsize*⟩ per i simboli delle espressioni, ⟨*subscriptsize*⟩ per gli indici e i pedici di primo livello e ⟨*subsubscriptsize*⟩ per gli indici e i pedici di secondo livello.

Queste dichiarazioni sono più utili di quanto si pensi, perché talvolta nelle classi standard è desiderabile apportare qualche modifica, specialmente per i corpi più piccoli e più grandi.

29.18.4 Simboli speciali

Qualunque segno di qualunque font può venire stampato prescindendo dalle sue caratteristiche particolari; il comando

`\symbol`⟨*indirizzo*⟩

permette di accedere a qualunque simbolo di qualunque polizza di caratteri mediante il suo ⟨*indirizzo*⟩; il carattere viene stampato prescindendo dalle sue caratteristiche speciali o dall'esistenza di un comando specifico per usarlo; l'unica questione è conoscerne l'indirizzo decimale, oppure ottale, oppure esadecimale. Attenzione: gli indirizzi si riferiscono alla polizza di font usati per la composizione, non ai caratteri introdotti o introducibili con la tastiera; per esempio le due parentesi graffe nella codifica di entrata hanno indirizzi 123 e 125 rispettivamente: se però si usano questi indirizzi con un font con codifica OT1, si ottiene – e ″, il che corrisponde perfettamente con quanto si può leggere nella tabella 18.3 della pagina 408.

Questo fatto è collegato al modo diverso di interpretare i caratteri immessi con la tastiera nel file sorgente `.tex` e soggetti alla codifica di entrata specificata con l'opzione fornita al pacchetto *`inputenc`* rispetto alla codifica usata per i font di uscita, specificata con l'opzione passata al pacchetto *`fontenc`*. Si veda più dettagliatamente questa questione nel capitolo 26.

Sia per i font codificati con un byte sia per quelli codificati con diversi byte (UTF-8, UNICODE) si può usare la *caret notation* che fa riferimento agli indirizzi esadecimali; per ogni cifra esadecimale (in caratteri minuscoli) dell'indirizzo bisogna premettere tanti segni "caret", `^`, quante sono le cifre che compongono l'indirizzo (sempre in numero pari, quindi con eventuali zeri iniziali). Perciò l'indirizzo esadecimale del segno `{` è 7B; la notazione `^^7b` è del tutto equivalente a indicare quel carattere tanto quanto `\char123`. Per gli indirizzi UNICODE si può fare questo esempio: la lettera Ž ha l'indirizzo UNICODE convenzionalmente indicato con `U+017D` e la sua notazione caret è `^^^^017d`, del tutto equivalente all'espressione `\v{Z}` del linguaggio di base LaTeX. Questa *caret notation* sembra una inutile complicazione, ma se i vari pacchetti non offrono nessun comando per usare un certo glifo di un font particolare, in particolare di un font UNICODE, e il sistema operativo non offre nessuna possibilità di inserire tali caratteri usando la tastiera e/o una tabella, allora la *caret notation* è l'unico modo per accedere a caratteri che non appartengono alla codifica ASCII. Un caso particolare si ha quando si predispongono i file dei pattern per lingue che non usano l'alfabeto latino senza diacritici, perché quando si creano i file di formato o lualatex legge tali file di pattern, non si possono ancora usare i pacchetti che definiscono i caratteri non ASCII, quindi la *caret notation* è l'unica che permette di accedere a tali caratteri. Insolito? Raro? Certo, ma i programmi di composizione del il sistema TeX fanno questo e molto altro, non usano soltanto il mark up LaTeX.

Capitolo 30

Aggiornamenti di LaTeX e dei pacchetti di servizio

Il linguaggio LaTeX e il software che lo interpreta è in continua evoluzione. È relativamente difficile rincorrere le varie modifiche che il LaTeX Project Team[1] vi apporta, nonostante sia sollecito nel pubblicare abbastanza spesso una *news letter* che contiene gli ultimi aggiornamenti.

Il LaTeX Team, con questo, o con un nome meno formale, esiste da dopo la conferenza del TUG di Cork nel 1991. Ora la parte che riguarda il nucleo di LaTeX e i *pacchetti di servizio*[2] viene seguita dal LaTeX Team, mentre la parte che riguarda le lingue è seguita dal Hyphen Team; questo secondo gruppo di lavoro si occupa di `babel` e `polyglossia` oltre che di tutti i file di descrizione delle lingue e dei file di pattern per le loro cesure.

Entrambi i gruppi di lavoro cercano di rendere tutti i programmi di composizione che si basano su LaTeX sempre più conformi alla codifica Unicode e questo richiede un grande lavoro, anche perché la gestione degli alfabeti con moltissimi caratteri, ben più dei 256 che possono essere contenuti nei file dei font Type 1, richiede tecniche particolari che inizialmente non erano state previste da TeX, anzi quando TeX è nato (in forma sperimentale nel 1978) Unicode non esisteva ancora e l'unica codifica universale era quella ASCII.

In questi anni sono nati anche altri programmi di composizione basati sui concetti di TeX, come per esempio ConTeXt, ma il LaTeX Team non se ne occupa ed esiste un altro gruppo che ne cura lo sviluppo. È nato anche LuaTeX che incorpora anche il linguaggio di scripting Lua, ma benché il funzionamento interno di LuaTeX sia diverso da quello di pdfTeX e XeTeX, accetta il file di formato che contiene le stesse macro usate per gli altri due interpreti e dà luogo

[1] Nel seguito indicato solo con la locuzione LaTeX Team.

[2] I pacchetti di servizio sono principalmente quelli che si trovano nella cartella `.../tex/latex/tools/` più pochi altri pacchetti come *graphicx*, *etoolbox*, *pict2e*, *amsmath*, e simili, che sono praticamente sempre usati dagli utenti.

al programma di composizione LuaLaTeX di cui si è parlato nei capitoli di questo testo.

In questo capitolo non si vuole replicare l'intero contenuto delle numerose *news letters* pubblicate dal 1994, anno di nascita di LaTeX 2$_\varepsilon$, perché possono essere lette direttamente, almeno con l'installazione di TeX Live, usando da terminale il comando texdoc seguendo la sintassi seguente:

`texdoc ltnews`⟨*numero*⟩

dove ⟨*numero*⟩ (sempre di due cifre: la prima lettera del 1994 si chiama 'ltnews01') è il numero progressivo della lettera. Per esempio un possibile comando potrebbe essere:

```
texdoc ltnews32
```

la cui esecuzione apre il visualizzatore di file PDF che mostra il testo della trentaduesima lettera.

Qui si cercherà di filtrare le informazioni più importanti per l'utente finale, lasciando i dettagli per le modifiche al sottostante linguaggio LaTeX 3 per gli sviluppatori di pacchetti e per i programmatori.

I paragrafi che seguono possono contenere delle ripetizioni di carattere cronologico; perché certi problemi col passare del tempo potevano richiedere ulteriori provvedimenti.

Le informazioni riportate qui di seguito partono dalla prima delle due *news letters* del 2018, anno in cui l'ammodernamento di LaTeX ha subito una radicale trasformazione. Tuttavia va ricordato che sia l'inclusione delle funzionalità di ε-TeX sia l'uso di pdfTeX come programma di interpretazione dei file sorgente, sono cose che risalgono alla fine degli anni '90. Di fatto anche il programma TeX, viene sostituito da pdfTeX, e lo stesso avviene per LaTeX, perché il formato preferito è sempre il PDF, non più il DVI; pdfTeX può produrre sia l'uscita in PDF, sia quella in DVI; l'utente continua ad usare i nomi dei programmi tex e latex, ma sotto il cofano è pdftex il programma che lavora.

Ricordo che XeLaTeX produce un file in formato DVI esteso (per gestire i caratteri Unicode), ma la sua uscita viene simultaneamente trasmessa al convertitore in formato PDF, quindi l'utente non si accorge che di mezzo c'è anche un passaggio attraverso il quale certe informazioni non passano. Esisterebbe anche una variante di LuaLaTeX capace di produrre un'uscita in formato DVI, ma ho il sospetto che gli utenti di questa variante siano pochissimi.

30.1 La lettera 28 dell'aprile 2018

La codifica dei file sorgente di default è ora UTF-8; non ci sarebbe più bisogno di specificare a pdfLaTeX di usare il pacchetto *inputenc* con l'opzione *utf8*, ma non succede niente di speciale se lo si fa; diventa invece necessario continuare ad usare quel pacchetto con altre codifiche diversa da *utf8*, essenzialmente per poter compilare "vecchi" documenti di anni precedenti.

I comandi `\counterwithin` e `\counterwithout` sono ora definiti nel nucleo di LaTeX; non è più necessario caricare l'uno o l'altro dei pacchetti *remreset* e *chngcntr*, come in passato.

Il nucleo di LaTeX conteneva un comando per controllare se una sequenza di controllo fosse già stata definita e lo faceva controllando attraverso `\csname` e `\endcsname` se la stringa di controllo fosse o non fosse già definita. Anche l'utente poteva usare queste due macro di delimitazione per svolgere diverse azioni particolari. Ma il difetto è che se la macro generata con questi delimitatori, non è definita, le viene attribuito lo stesso significato di `\relax` e il confronto tramite il test con `\ifx` era vero se la sequenza di controllo generata era equivalente, appunto, a `\relax`. Tutto ciò poteva portare a falsi negativi, perché di fatto la sequenza di controllo indefinita era definita come `\relax`. Ora il test viene eseguito con i delimitatori `\ifcsname` e `\endcsname` di ε-TeX, che analizza la control sequence senza definirla col valore di `\relax`, e il problema non si presenta più. Il problema si pone anche con l'argomento si `\show`, che però è stato risolto diversamente in aggiornamenti successivi (vedi più avanti).

Il pacchetto *array*, utile per estendere la funzionalità dei vari ambienti per creare tabelle e matrici, ora contiene i descrittori di colonna `w` e `W`, per costruirle con colonne della stessa larghezza.

Il pacchetto *multicol* è stato aggiustato nel senso che il comando `\columnbreak` non produce gli inconvenienti che prima talvolta produceva.

30.2 La lettera 29 del dicembre 2018

Il nucleo di LaTeX con l'introduzione della codifica UTF-8 di default, ha richiesto qualche correzione perché alcuni indirizzi di glifi erano errati; ma sono stati anche introdotti i nomi corretti delle macro destinate a usare i caporali: `\guillemetleft` e `\guillemetright`, perché i nomi precedenti (conservati per retrocompatibilità) facevano riferimento alla parola di origine francese *guillemot* che indica un uccello marino.

Il comando `\verb*` e l'ambiente *verbatim** sono stati corretti per evitare che uno spazio immediatamente seguente il comando, o l'apertura dell'ambiente, venisse interpretato come carattere delimitatore del materiale da scrivere in modo verbatim.

I comandi `\label` e `\index` potevano produrre dei salti di pagina in posti dove erano vietati, per esempio fra due comandi di sezionamento consecutivi, per esempio un comando `\section` immediatamente seguito da un comando `\subsection`. Questo errore è stato eliminato e se si vuole inserire un salto pagina in quella posizione insolita bisogna usare espressamente un comando `\pagebreak`.

I comandi `\thinspace`, `\phantom` e `\smash` sono da usare strettamente in modo testo, ma quei comandi inseriti all'inizio di un capoverso non facevano il loro mestiere e il risultato era una riga bianca prima del capoverso. Questo

errore è stato corretto anche in *amsmath*, che ridefiniva questi commandi ma, di nuovo, senza la correzione inserita nel nucleo di LaTeX.

Il pacchetto *trace* serve per scrivere nel file `.log` tutto quello che l'interprete fa, e scrive migliaia di righe, molte delle quali non sono di nessun interesse; perciò è stato modificato per disattivare il tracciamento prima di eseguire certi blocchi di codice, per riprenderlo in automatico alla fine dell'esecuzione di questi blocchi; tuttavia questa prima modifica cancellava troppo, e la situazione è stata corretta. Il tracciamento è un'operazione che l'utente finale fa raramente, ma spesso è l'unica maniera che l'utente finale ha per scoprire la causa di certi errori. Semplificare il tracciato è molto importante, quindi queste modifiche sono molto utili.

Il pacchetto *xr* è poco conosciuto; serve per eseguire riferimenti incrociati con oggetti che si trovano in altri documenti. Ovviamente lavora servendosi delle funzionalità di *hyperref*. Questa interazione in certi casi produceva errori di basso livello e sono state apportate le modifiche necessarie per eliminare questi errori.

L'ambiente *multicols** in certe circostanze poteva perdere delle parti di testo; l'errore è stato eliminato. Nello stesso modo è stato eliminato il comportamento imprevedibile del comando `\docolaction` del pacchetto *multicol*, che fa cose diverse a seconda che venga eseguito quando si sta componendo nella prima o nell'ultima colonna di una pagina, o in una colonna intermedia.

Con il pacchetto *array*, quando si colorano le celle di una tabella, il colore poteva espandersi fuori dalle celle. L'errore è stato corretto.

Sempre in *array* nel campo di descrizione delle colonne potevano venire aggiunti tramite i suoi descrittori specifici `>` e `>` dei comandi o delle dichiarazioni fragili, che tendevano a produrre errori irreparabili; la causa del problema è stata individuata e corretta.

30.3 La lettera 30 dell'ottobre 2019

In linea con l'adozione della codifica UTF-8 predefinita, ora comandi come `\label` possono ricevere anche caratteri non ASCII; per esempio è lecita una scrittura del genere: `\label{eq:größer}`. Tuttavia continuano a essere vietati nomi di comandi formati con stringhe letterali contenenti caratteri diversi dalle lettere ASCII.

Invece i nomi dei file da da leggere con `\input` o `\include` possono contenere caratteri Unicode; era vietato prima di questa modifica. Analogamente i nomi di file per il comando `\includeonly` possono ora non solo contenere caratteri non ASCII ma anche le virgole che separano i vari nomi possono essere circondate da spazi, inclusi i fine riga del file sorgente.

Le funzionalità dell'ambiente *filecontents*, anche con le estensioni del pacchetto *filecontents*, sono diventate parte integrante del nucleo di LaTeX, e rendono inutile l'uso del pacchetto; inoltre sono rimosse tutte le limitazioni dell'ambiente. Il nuovo ambiente accetta delle opzioni: *overwrite*, che serve per riscrivere un

file già presente nella cartella di lavoro, ma controlla che non si riscriva il file `\jobname.tex`, altrimenti si produrrebbero danni fatali a questo file; l'opzione *noheader* elimina dal file il messaggio iniziale (composto come commento LaTeX) che normalmente inserirebbe; l'opzione *nosearch* serve per escludere la ricerca dei file da riscrivere in cartelle diverse da quella locale.

Molti comandi sono stati resi robusti, inclusi `\begin` e `\end`; quest'ultimo era particolarmente delicato e ha richiesto un trattamento speciale.

Nell'usare la codifica UTF-8 di default, apparentemente sono risultati errati i *code point* dei simboli ottenibili con `\textlangle` e `\textrangle`; problema risolto.

Il comando `\InputIfFileExists` è stato corretto; in alcune circostanze poteva sbagliare il test e caricare altri file.

I pacchetti *fncylab* e *varioref* permettevano di usare il nome interno `\p@...` di un contatore LaTeX per accedere al suo contenuto; fornivano anche comandi come `\labelformat` e `\Ref`; *varioref*, da parte sua, con il comando `\vref` scriveva il riferimento in modo particolare. Tutti agivano in modo particolare quando veniva usato il comando `\refstepcounter`. Ora tutte queste operazioni sono diventate stabili e in particolare i comandi `\labelformat` e `\Ref` sono migrati nel nucleo di LaTeX.

Il pacchetto *array* permette di definire nuovi descrittori di colonne per tabelle e matrici; sarebbero evidentemente vietate le ridefinizioni dei descrittori originali; il messaggio che accompagnava queste ridefinizioni illecite non era funzionante correttamente; ora l'errore è stato corretto.

Il pacchetto *multicol* è stato esteso per gestire meglio il bilanciamento delle colonne nell'ultimo moncone del suo testo; ora è anche possibile stabilire il minimo numero di righe da inserire nelle ultime colonne, quando le righe che dovrebbero contenere sono poco numerose.

Il pacchetto *varioref* è stato aggiornato per gestire meglio anche le funzionalità di *cleveref* in connessione con *hyperref*. Tra le altre cose ora gestisce anche l'arabo.

Il pacchetto *xr* per i riferimenti esterni è stato esteso anche alle citazioni di testi elencati nelle bibliografie di altri documenti; tuttavia non funziona con *hyperref*; per farlo funzionare anche in questo caso bisogna usare il pacchetto *xr-hyper*.

Il pacchetto *amsmath* ora dispone oltre ai comandi `\overset` e `\underset` anche del nuovo comando `\overunderset` per inserire informazioni sia sopra sia sotto gli operatori binari.

30.4 La lettera 31 del febbraio 2020

Il programma LuaLaTeX ora usa LuaHBTeX; cioè include la libreria HarfBuzz, che serve per gestire le infinite caratteristiche dei font OpenType. Sostanzialmente il pacchetto *fontspec* si occupa di operare tutte le impostazioni necessarie attraverso le funzionalità di quella libreria.

Il tempo di caricamento di LuaLaTeX ora è fortemente ridotto perché il codice di *expl3* non viene più caricato quando si lancia il programma, ma è già inserito nel file di formato; questo diventa particolarmente importante per poter usufruire delle poderose funzionalità di *xparse*; inoltre il pacchetto *fontspec* è tutto scritto in linguaggio LaTeX 3. Perciò non è più necessario caricare il pacchetto *expl3* se non per compilare vecchi documenti che ne facevano uso.

Le funzionalità del comando `\fontshape` dell'insieme di comandi per la selezione dei font (a cui ci si riferisce con la sigla NFSS) ora permette di gestire la forma dei caratteri in modo molto complesso e con sigle nuove, uniformate fra molti pacchetti di gestione dei font; sono disponibili per esempio i codici `scit` per il maiuscoletto corsivo, `scsl` per il maiuscoletto inclinato, `sw` per i caratteri *swash* (caratteri in generale corsivi, ma con le maiuscole leggermente calligrafiche). Non solo ma il comando `\fontshape` qualche volta aggiunge il suo codice ad un altro di descrizione della forma già in vigore; per esempio `\fontshape{sc}\fontshape{it}` è equivalente a `\fontshape{scit}`.

Analogamente le funzionalità di `\fontseries` sono estese per dichiarare molti più pesi (o forze) dei caratteri da leggerissimo a nerissimo.

Le modifiche alla gestione delle forme e delle serie possono essere personalizzate. Comunque il LaTeX Team ritiene che queste cose siano di interesse per i creatori di classi o di pacchetti, ma ammettono che i comandi di basso livello sono così facili da usare che molti utenti finali se ne possono servire; tuttavia non indicano una documentazione da consultare per evitare di perdere tempo a cercare informazioni sparse nella rete. Chi scrive ha trovato un po' di documentazione nel capitolo `s`[3] della guida `source2e.pdf` che si legge con il comando `texdoc source2e`.

La famiglia di font da usare si potrebbe cambiare con `\fontfamily{`⟨*famiglia*⟩`}`, ma questo cambio potrebbe essere legato alla codifica; la lettera 31, propone questo esempio: si supponga di voler cambiare famiglia quando si scrive in caratteri latini usando la famiglia Montserrat-LF con codifica *T1* e si voglia passare al greco con codifica *LGR* usando però il font IBMPlexSans-TLF; in questo caso si dichiari una differente famiglia mediante il comando:

```
\DeclareFontFamilySubstitution{LGR}{Montserrat-LF}{IBMPlexSans-TLF}
```

Il pacchetto ausiliario *texcomp* non è più necessario, perché tutti i comandi per usare i font di questo pacchetto sono già tutti disponibili nel nucleo di LaTeX. Ciò nonostante se si carica il pacchetto *textcomp*, non succede nulla di grave. La nuova situazione ha anche una gestione delle sostituzioni più intelligente, nel senso che se un segno manca in una data famiglia, esso viene sostituito con un segno equivalente della stessa famiglia, non più della famiglia Computer Modern. Sembra che questa nuova impostazione sia particolarmente visibile e utile quando si usano le cifre minuscole con `\oldstylemums`: in tondo graziato si ottiene 12345, con un font senza grazie si ottiene 12345, e con un font teletype si ottiene 12345.

[3] Sì, i capitoli di questo documento sono "numerati" con lettere minuscole.

I comandi `\ProvidesClass` e `\ProvidesPackage` vengono usati dai programmatori che creano classi o pacchetti, ma sarebbe bene che li usassero anche gli utenti finali quando generano una nuova classe o si costruiscono un pacchetto di macro personali; sarebbe utile usare anche `\ProvidesFile` all'inizio di ogni altro file, in modo da registravi anche la data e lo scopo. Tuttavia il nome del file non poteva contenere altro che caratteri ASCII così come l'argomento opzionale di questi comandi; adottando di default la codifica UTF-8, ora questi argomenti possono contenere qualsiasi carattere.

Il comando `\textbackslash` ora è di nuovo robusto, visto che con il precedente aggiornamento era diventato fragile all'interno di comandi del tipo `\raggedright` e compagni.

Anche i delimitatori degli ambienti matematici, pur essendo robusti, talvolta davano problemi; sono stati corretti sia nel nucleo di LaTeX sia in *amsmath*.

Come noto, i programmi di composizione del sistema TeX basati su LaTeX possono usare contemporaneamente solo 16 flussi di scrittura di file; LuaLaTeX può usare più di 16 flussi e il problema che presentava `\filecontents` per esaurimento di flussi disponibili, non esiste più; rimane però con pdfLaTeX e XeLaTeX.

Sono stati resi robusti i comandi dei pacchetti *color* e *graphicx*.

È stato corretto il pacchetto *multicol* che usava malamente il comando `\maxdepth` al posto di `\@maxdepth`.

Come accennato in una lettera precedente, il pacchetto *multicol* talvolta presentava il difetto di perdere del testo. Questo problema era collegato con l'uso delle (molte) scatole di cui *multicol* ha bisogno e di come il programma di composizione le alloca; se altri pacchetti ne usavano molte, c'era il rischio che venisse coinvolta la scatola 255, riservata all'uso dell'operazione di raccolta del materiale parzialmente impaginato prima di assemblarlo per spedirlo al file di uscita. Ora *multicol* usa solo scatole oltre quella numerata 255, perché, le scatole disponibili oltre a quel numero sono quasi illimitate (oltre 32 000).

30.5 La lettera 32 dell'ottobre 2020

Ora le funzionalità di *xparse* sono quasi tutte migrate nel nucleo di LaTeX. Le uniche funzionalità che richiedono ancora l'uso del pacchetto *xparse* sono i descrittori degli argomenti `g`, `G`, `l`, e `u`; i primi due sono "deprecati"[4] ma sono ancora disponibili coll'uso del pacchetto per una questione di retrocompatibilità. Pertanto ora con tutti i motori di composizione basati su LaTeX sono disponibili i comandi:

- `\NewDocumentCommand`, `\RenewDocumentCommand`, `\ProvideDocumentCommand`, `\DeclareDocumentCommand`

[4] Si tratta di descrittori di argomenti facoltativi delimitati da graffe; il loro uso è in palese contrasto con la regola generale di LaTeX, per la quale gli argomenti delimitati da graffe sono quelli obbligatori.

- `\NewExpandableDocumentCommand`, `\RenewExpandableDocumentCommand`, `\ProvideExpandableDocumentCommand`, `\DeclareExpandableDocumentCoommand`
- `\NewDocumentEnvironment`, `\RenewDocumentEnvironment`, `\ProvideDocumentEnvironment`, `\DeclareDocumentEnvironment`
- `\BooleanTrue`, `\BooleanFalse`
- `\IfBooleanTF`, `\IfBooleanT`, `\IfBooleanF`
- `\IfNoValueTF`, `\IfNoValueT`, `\IfNoValueF`
- `\IfValueTF`, `\IfValueT`, `\IfValueF`
- `\SplitArgument`, `\SplitList`, `\TrimSpaces`, `\ProcessList`, `\ReverseBoolean`
- `\GetDocumentCommandArgSpec`, `\GetDocumentEnvironmentArgSpec`

Alcuni di questi comandi hanno un significato evidente; per il resto serve sempre la documentazione del pacchetto *xparse*. La cosa nuova rispetto ai comandi tradizionali di LaTeX 2ε, è che non bisogna specificare gli argomenti con un numero, che indica quanti argomenti il comando o l'ambiente può ricevere, ma bisogna specificare una lista di descrittori che specificano le caratteristiche di ogni argomento. Inoltre gli argomenti del comando di apertura dell'ambiente possono venire usati anche all'interno dei comandi di chiusura; con i comandi tradizionali essi potevano essere usati solo nei comandi di apertura.

Le operazioni del pacchetto *calc* sulle lunghezze possono essere usate anche all'interno dell'ambiente *picture*; così si può scrivere una cosa come

```
\put(0.5\textwidth, 0.5\textheight){\circle*{1ex}}
```

e il risultato delle espressioni dimensionali viene usato dai comandi grafici dell'ambiente *picture* come dei valori assoluti, indipendenti dal valore di `\unitlength`. Questa funzionalità può essere molto utile per certe applicazioni ma va usata con cognizione di causa; infatti i soliti comandi di *picture* non funzionano con le espressioni eseguite mediante `\dimexpr`, perché usano come coordinate solo valori indicati con numeri reali, quindi non mediante dimensioni.

I nomi dei file da immettere con `\include` e da elencare nella lista usata come argomenti di `\includeonly` possono ora contenere spazi.

Quando si compone con le dichiarazioni che non giustificano il testo o nei corrispondenti ambienti, parole lunghissime che compaiono nella penultima riga del testo possono provocare una riga vuota; vi si rimedia specificando `\finalhyphendemerits=0`.

È consigliabile specificare un valore non nullo per `\baselineskip` quando si usano i comandi `\textsupercript` e `\textsubscript` perché servono a *hyperref* per determinare la grandezza della scatola che contiene il collegamento ipertestuale.

Si raccomanda di non usare più `\seriesdefault` ma, per essere compatibili con la nuova funzionalità delle impostazioni della serie dei font, di usare `\DeclareFontSeriesDefault`.

Alcuni comandi insoliti, come per esempio il gruppo

```
\renewcommand\ttdefault{lmvtt}
```

```
\DeclareFontSeriedDefault[tt]{md}{lm}
\DeclareFontSeriesDefault[tt]{bf}{bm}
```

per commutare dalla famiglia dei font graziati a quella dei font teletype, qualche volta non davano i risultati desiderati se coinvolgevano la serie media e la serie nera. Ora la cosa è stata sistemata.

Con le equazioni appoggiate a sinistra, quando si specifica l'opzione *fleqn*, LaTeX inserisce uno spazio elastico che si chiama `\mathindent` in modo che le equazioni fuori testo non si appoggino al margine sinistro ma siano un poco rientrate; questo rientro elastico può ridursi fino a zero se l'equazione è molto larga. Inoltre in ***amsmath*** viene inserito uno spazio fra il numero dell'equazione e la formula, in modo che questa non si scontri con il suo numero identificativo. Queste funzionalità ora sono presenti anche nelle classi standard.

LaTeX ha ereditato dal formato Plain di TeX i comandi `\llap` (*left overlap*) e `\rlap` (*right overlap*) che producono scatole di larghezza nulla, col contenuto che sporge rispettivamente a sinistra o a destra; LaTeX ora dispone anche della scatola `\clap` sempre di larghezza nulla, ma col contenuto che sporge simmetricamente da entrambi i lati.

Sono stati aggiunti i test per controllare le date del formato, delle classi e dei pacchetti. Le sintassi di tali comandi sono ora:

```
\IfFormatAtLeastTF{⟨data⟩}{⟨vero⟩}{⟨falso⟩}
\IfClassAtLeastTF{⟨classe⟩}{⟨data⟩}{⟨vero⟩}{⟨falso⟩}
\IfPackageAtLeastTF{⟨pacchetto⟩}{⟨data⟩}{⟨vero⟩}{⟨falso⟩}
```

Questi comandi confrontano la ⟨*data*⟩ specificata per il formato o per la ⟨*classe*⟩ o per il ⟨*pacchetto*⟩ specifici con la data contenuta nei loro comandi `\Provides...` ed eseguono il codice ⟨*vero*⟩ solo se la data sotto esame è posteriore o uguale alla ⟨*data*⟩ specificata, altrimenti eseguono il codice ⟨*falso*⟩.

Con LaTeX 2_ε è difficile controllare quale sia la definizione di un comando protetto. Il comando avrebbe potuto essere stato ridefinito da una classe o da un pacchetto oppure l'utente finale potrebbe aver definito una macro che non fa quello che sperava, in realtà a causa di un errore dell'utente, ma che passa i test del programma interprete; ora il significato dei comandi robusti (cioè protetti) viene mostrata correttamente se si usa il comando `\ShowCommand`; per esempio, se si da il comando `\ShowCommand\frac`, nel terminale apparirà il seguente testo (seguito da altre righe che qui non interessano):

```
>␣\frac=robust␣macro:
->␣\protect␣\frac␣␣␣.

>␣\frac␣=\long\macro:
#1#2->{\begingroup␣#1\endgroup␣\@@over␣#2}.
```

che corrisponde esattamente al testo di sostituzione della definizione di `\frac`.

Ora nell'argomento di `\typeout` possono anche apparire comandi `\par` espliciti o righe vuote. Prima questi comandi e queste righe producevano errori.

Certi comandi di spaziatura matematica definiti nel pacchetto *amsmath* sono migrati nel nucleo di LaTeX. Si tratta dei comandi seguenti

		comandi	matematica	testo
\,		\thinspace	$x\,x$	x x
\!		\negthinspace	$x\!x$	xx
\:	\>	\medspace	$x\:x$	x x
		\negmedspace	$x\!x$	xx
\;		\thickspace	$x\;x$	x x
		\negthickspace	$x\!x$	xx

Nel pacchetto *array* ci sono due comandi poco noti \firsthline e \lasthline che svolgono un ruolo simile a quello di \toprule e \botmrule del pacchetto *booktabs*; prima producevano un allargamento indesiderato della tabella; ora questo difetto è stato eliminato.

Il comando \externaldocument del pacchetto *xr* ora accetta nomi di file che contengono spazi; si comporta come \include e \includeonly.

Gli ambienti *aligned* e *gathered* presentavano un difetto; essi accettano fra parentesi quadre un parametro facoltativo di incolonnamento che permetteva di allineare verticalmente il loro contenuto rispetto al contenuto degli ambienti nei quali sono usati; però, se il primo token del loro contenuto è una parentesi quadra aperta, le macro che gestiscono il campo facoltativo intervenivano a sproposito; ora dopo 25 anni[5] di questo baco, finalmente esso è stato eliminato.

Da anni il pacchetto *luatex-math* provvedeva a ridefinire i comandi \frac, \genfrac e l'ambiente *subarray* per usare codice Lua; ora questo codice è stato integrato nel pacchetto *amsmath*.

Anche il pacchetto *babel* ha subito molte modifiche, ma non se ne occupa il LaTeX Team, bensì lo Hyphen Team; però la lettera 32 invita il lettore a verificare nella documentazione di *babel* le novità di solito poco conosciute di questo potentissimo gestore delle lingue; si usi `texdoc babel` per accedere alla documentazione completa di *babel*.

30.6 La lettera 33 del giugno 2021

Un'importante modifica riguarda la gestione dei nomi dei file; ora questi nomi possono contenere spazi (già consentiti da alcuni anni; ma secondo chi scrive sono sempre sconsigliabili), ma ora possono anche contenere diversi punti, non solo quello che separa il nome dall'estensione (per esempio è gestibile un file nominato `il mio file del 2022.10.29.tex`) e caratteri non contenuti nella codifica ascii. Questa modifica risolve una quantità di problemi che riguardano il nuovo modo di gestire tutti i file `.tex` con la codifica Unicode.

Per quanto riguarda i nomi dei file da introdurre con \include, precedentemente non si doveva specificare l'estensione, perché l'estensione `.tex`

[5] Osservazione contenuta nella lettera 32.

veniva aggiunta automaticamente. Ora se l'utente mette l'estensione scrivendo `\include{ilmiofile.tex}`, LaTeX non cerca più di caricare il file `ilmiofile.tex.tex`, come succedeva prima, ma riconoscendo che l'estensione `.tex` esiste già, non ne aggiunge un'altra.

Una conseguenza favorevole della nuova gestione dei file è che se un file contiene nel suo nome un backslash, (chi farebbe mai una cosa del genere? o forse può succedere con gli utenti che usano una piattaforma Windows, nonostante anche per loro sia specificato che i percorsi fra le cartelle devono essere indicati con la barra e non con la barra inversa?) viene gestito in modo da togliere il backslash; l'esempio riportato nella news letter 33 dice: se si dispone di un file di nome `\sqrt{2}`, LaTeX gestisce questo nome trasformandolo in `sqrt␣{2}` e consentirebbe di importare un file il cui nome completo sarebbe `sqrt␣{2}.tex`, presumibilmente non esistente, cosicché verrebbe emesso il messaggio File not found. questo non è consolante, ma se LaTeX non avesse tolto il backslash sarebbe entrato in un ciclo infinito, il che sarebbe stato molto peggio.

Già da alcune versioni il nome di un file gestito con `\filecontents` accettava nomi contenenti caratteri codificati in Unicode, ma non ne consentiva la riscrittura; questo impedimento è stato rimosso.

Per la gestione dei font le varie dichiarazioni per famiglia, serie, forma e corpo potevano essere specificate in qualunque ordine, ma con le enormi estensioni introdotte con la versione di LaTeX di pochi anni fa, si è ritenuto di ritardare le dichiarazioni di serie e di forma a dopo che la famiglia è nota; quindi quelle scelte sono ritardate fino all'esecuzione di `\selectfont`, proprio per evitare che vengano eseguite sostituzioni errate, visto che molte di quelle innumerevoli scelte dipendono dalla famiglia dei font.

Per la gestione della code page relativa ai caratteri del blocco di caratteri *Latin Extended Advanced* le funzionalità di LaTeX sono ora disponibili per l'immissione diretta (tastiera permettendo) di quei caratteri: se la tastiera lo consente si può inserire direttamente, per esempio, il glifo ṃ, senza bisogno di ricorrere alla scrittura `\d{m}`; mediante la tastiera del Mac è possibile farlo attraverso la finestra del visualizzatore della tastiera. Con altre piattaforme bisogna ricorrere alla finestra dei caratteri Unicode.

Oltre ai soliti tratti orizzontali ottenibili con le legature `-, --, ---`, che producono -, –, —, sono ora disponibili i comandi `\textnonbreakinghyphen`, `\textfiguredash` e `\texthorizontalbar`; con pdflatex, usato per comporre questo testo, essi producono i segni -, – e —, ma con xelatex e lualatex essi producono caratteri veri, non approssimati con quanto è disponibile con i font codificati con 8 bit.

Il carattere ottenibile con il comando `\textasteriskcentered` serve in molti casi, in particolare come segno non alfanumerico per le note. Con font in codifica `T1` il carattere richiesto talvolta non è disponibile; per evitare il messaggio di "carattere non esistente" e un segno strano nel file composto, ora questo asterisco centrato mancante viene reso con un normale asterisco abbassato e ingrandito. Chi scrive preferisce approssimarlo con il glifo matematico `\star`, ma anche con la sua preferenza si tratta sempre di una approssimazione.

Da alcuni anni i font della collezione *Text Companion* codificata `TS1` sono già disponibili nel nucleo di LaTeX, e non è più necessario caricare il pacchetto *`textcomp`*. Ma quali glifi sono contenuti in questa collezione? E tutti i glifi sono disponibili con ogni famiglia? Il problema è stato risolto mappando i glifi mancanti su quelli della collezione TeX-Gyre simili per forma. In questo modo le cose sono tornate relativamente semplici, perché i font della collezione TeX-Gyre sono molto completi. L'utente ora non dovrebbe avere più noie con i caratteri della collezione Text Companion.

Per quel che riguarda la bibliografia ora il comando `\nocite` può essere messo dovunque, sebbene il preambolo sembri essere il luogo più appropriato. Prima era solo consentito inserirlo nel corpo del documento, vale a dire dire dopo `\begin{document}`

Per le note marginali è bene ricordare che ora il nucleo di LaTeX inserisce correttamente il comando di fine capoverso nel corpo del testo della nota.

Il segno `\\` ora dentro le tabelle normali e lunghe funziona come ci si aspetta. Prima, nelle celle codificati con i codici di posizione che implicano più righe in una stessa cella era ambiguo se si riferisse un "a capo" per il testo della cella, oppure se indicasse un fine riga della tabella. Ora la cosa è sistemata e indica il fine riga del testo della cella.

L'ambiente *`longtable`* poteva dare dai problemi se nella pagina dove cominciava la tabulazione era presente anche un oggetto flottante. Ora non può più succedere, ma la correzione potrebbe dare fastidio nella compilazione di vecchi documenti; quindi, nel caso, si ricorre al *rollback* della versione di *`longtable`* del gennaio 2020.

30.7 La lettera 34 del novembre 2021

Sono stati aggiunti due nuovi comandi, `\NewCommandCopy` e `\ShowCommand`. Il primo serve per creare una copia di un comando esistente in un nuovo comando che, volendo può essere modificato a piacere; `\ShowCommand` serve per mostrare nella console le caratteristiche di un comando e la sua definizione; questo secondo comando è particolarmente utile per mostrare il significato dei comandi robusti.

Due nuovi comandi, `\ClassNote` e `\PackageNote` sono ora disponibili, sono una via di mezzo fra i comandi esistenti che terminano con la parola `Warning` e quelli che terminano con `Info`. Essi mostrano nella console e scrivono nel file `.log` una annotazione che non è un avviso, ma che comunque identifica un messaggio che arriva da uno specifico pacchetto o da una specifica classe. La sintassi è simile a quella dei comandi terminanti in `Info`.

Il nuovo comando `\ShowFloat` si riferisce ad un ambiente flottante. Serve per vedere che cosa c'è dentro uno dei registri della coda dei float; un registro di tale coda ha il nome `\bx@`⟨*identificatore*⟩, dove l'identificatore di una o due lettere maiuscole. Il comando `\ShowFloat{`⟨*identificatore*⟩`}` permette di sapere che cosa è contenuto in quel registro.

I nuovi comandi `\counterwithin` e `\counterwithout` permettono di aggiungere o togliere nuovi contatori dalla liste subordinate ad altri contatori,

permettono anche di specificare come deve essere indicato il particolare contatore, se in cifre arabe, numeri romani, lettere maiuscole o minuscole. Da tempo il comando `\IfPackageLoadedTF` è disponibile per sapere se un dato pacchetto sia stato già caricato, ora si aggiunge il comando `\IfPackageLoadedWithOptionsTF` che con la sintassi

`\IfPackageLoadedWithOptionsTF{`⟨*pacchetto*⟩`}{`⟨*opzioni*⟩`}{`⟨*vero*⟩`}{`⟨*falso*⟩`}`

permette di scoprire se le date ⟨*opzioni*⟩ sono definite nel ⟨*pacchetto*⟩ e permette di agire di conseguenza eseguendo i comandi contenuti in ⟨*vero*⟩ o quelli contenuti in ⟨*falso*⟩

Ora fra i font per pdfLaTeX e per la coppia xelatex/lualatex i comandi per usare i font corsivi diritti operano direttamente, senza sostituire quelli Unicode con quelli a 8-bit.

Quando si definisce un nuovo comando, qualsiasi macro prevista a questo scopo controlla che il nome del nuovo comando non sia ancora stato usato. Quando si definisce un nuovo ambiente `foo` (che agisce mediante il comando di apertura `\foo` e il comando di chiusura `\endfoo`, il precedente comando `\newenvironment` controllava che né il comando di apertura né quello di chiusura fossero già stati usati. Il nuovo comando `\NewDocumentEnvironment` controllava solo la preesistenza di `\foo`, ma non verificava la preesistenza di `\endfoo`. Con la nuova versione descritta in questa news letter, ora anche il nuovo comando verifica la preesistenza del comando di chiusura.

È stato corretto il comando `\contentsline` usato da LaTeX per comporre l'indice generale, aveva un conflitto con l'uso di *hyperref* che si manifesta solo in un secondo tempo, quando l'utente decideva di non usare più i riferimenti incrociati; alla prima compilazione dopo questa decisione la compilazione leggeva il file prodotto nella compilazione precedente creata prima di assumere quella decisione.

Il comando `\TrimSpaces` elimina gli spazi attorno agli argomenti di certi comandi; talvolta questi spazi producono effetti indesiderati, specialmente in matematica.

All'interno dell'ambiente *multicols* è usabile il comando `\columnbreak` che funziona come `\pagebreak` con i 5 possibili valori di una cifra da 0 a 4, per specificare la "forza" del comando. Inoltre l'ambiente dispone del comando `\newcolumn` che termina la colonna corrente similmente a come `\newpage` termina la pagina corrente.

30.8 La lettera 35 del giugno 2022

Il meccanismo dei *marks* per raccogliere informazioni da inserire nelle pagine completamente composte alla routine che accoda le pagine al file di uscita funziona abbastanza bene con la maggioranza dei casi "normali", che non richiedono pagine particolarmente complesse. Ma nel tempo sono state create estensioni sia mediante nuove classi, sia mediante pacchetti per comporre pagine complesse.

Il nuovo nucleo contiene un nuovo insieme di *marks* del tutto autonomi e indipendenti che possono essere impostati individualmente e possono essere usati anche in contesti differenti dalla pagina. Esistevano un pacchetto del 2021, precursore di questa innovazione, e il pacchetto *titleps* per disegnare stili di pagina complessi; le loro testatine e piedini corrispondono più o meno alla disposizione ottenibile con *fancyhdr*, ma tutti e sei i campi, tre dei piedini e tre delle testatine, sono gestiti attraverso i nuovi *marks*. La documentazione per l'uso di questo nuovo insieme di mark leggibile con `texdoc ltmarks-doc`.

Il LaTeX Team ha aggiunto al nucleo di LaTeX il meccanismo per definire le opzioni del tipo "chiave=valore", i comandi per gli autori sembrano più semplici di quelli definiti dai pacchetti preesistenti. Il meccanismo "chiave=valore" permette di fare cose impensabili usando il semplice LaTeX 2_ε e dovrebbe essere usato più spesso. La documentazione è quella del pacchetto *l3keys2e*, leggibile con `texdoc l3keys2e`, ma il pacchetto non si deve più usare perché il suo codice fa ora parte del nucleo di LaTeX.

Il codice che realizza le funzioni `\fpeval` e `\inteval` (per eseguire calcoli con numeri decimali in virgola mobile e, rispettivamente, calcoli con numeri interi) non richiede più l'uso del pacchetto *xfp* (che rimane utile come documentazione). Questo pacchetto ridiviene utile solo in quei pochissimi casi in cui certi descrittori degli argomenti sono ancora sperimentali e potrebbero non venire mai aggiunti al nucleo di LaTeX. In occasione di questa estensione al nucleo di LaTeX sono state aggiunte anche le nuove funzioni `\dimeval` e `\skipeval` per eseguire calcoli sulle lunghezze rigide e su quelle elastiche. Si noti che anche `\fpeval` permette di eseguire calcoli su queste grandezze, assumendone come valore il loro unico o principale valore numerico espresso in punti tipografici, ma a differenza dei comandi `\the`, o `\dimexpr` perdono la natura dimensionale; tuttavia se si aggiunge `pt` (o `\p@`) subito dopo la funzione, i calcoli con le grandezze si possono fare ugualmente; per esempio si può scrivere

```
\newlength\totalheadheight
\totalheadheght=\fpeval{\headheight+\headsep}pt
```

I comandi `\numexpr`, `\dimexpr` e `\glueexpr` fanno già parte dei motori di composizione tex, etex, pdftex, xetex, e (forse) di luatex, ma soffrono il difetto di considerare numeri binari di 30 o 31 bit, corrispondenti a circa una decina di cifre decimali, con la metà sempre assegnata alla parte fratta (calcoli a virgola fissa). Con `\fpeval` i calcoli sono eseguiti in base decimale, non in base binaria, e possono essere eseguiti con un numero anche molto alto di cifre; `\fpeval` è prezioso; `\inteval` lo è forse un po' di meno, ma è più veloce di `\fpeval` quando si devono calcolare espressioni numeriche fra operandi interi.

Insieme ad altre funzionalità, si è provveduto a correggere alcuni bachi; uno, in particolare, riguarda il comando `\newcolumn` che, se emesso mentre la composizione si trova nello stato di "vertical mode", produceva effetti strani ed errori; l'errore è stato corretto.

30.9 La lettera 36 del novembre 2022

Questa lettera annuncia importanti passi avanti, non solo nella correzione dei pacchetti di pertinenza del LaTeX Team, cioè quelli che sono conservati nelle cartelle `/base` e `/tools` dell'albero principale di cartelle di TeX Live, ma anche nella correzione del nucleo di LaTeX.

Fra i miglioramenti dei codici appaiono i seguenti.

- Impostazioni per l'uso dei font EC maiuscoletti con codifica T1.
- Impostazioni per l'uso dei font EC lineari in corpo piccolo con codifica T1.
- Miglioramento della gestione della serie nera nel caso che il file `.fd` sia difettoso.
- Individuazione degli ambienti *minipage* annidati.
- Irrobustimento di comandi nella gestione delle opzioni dei pacchetti.
- Miglioramento dell'integrazione del pacchetto *l3docstrip* con *docstrip*.
- Miglioramento nell'efficienza del *callback* di luatex
- Realizzazione dell'ordinamento basato su regole per i gestori del *callback* di luatex.
- Correzione del bug connesso al comando `\mathcolor` introdotto recentemente per colorare parti di espressioni matematiche.
- Miglioramento introdotto nel pacchetto *array* per il riconoscimento di celle di tipo `m` contenenti una sola riga.

Un grosso passo avanti nel nucleo di LaTeX è costituito dalla modifica del pacchetto *ltcmnd* che ora accetta il modificatore `=...` nel raccogliere gli argomenti. In questo modo LaTeX sa che tali argomenti vanno trasmessi al sottostante codice come opzioni della forma "chiave=valore". Per esempio, il comando `\caption` potrebbe venire definito così

```
\DeclareDocumentCommand\caption
        {s ={short-text}+O{#3} +m}
        {...}
```

che significa che se il l'argomento facoltativo non contiene espressioni del tipo "chiave=valore" esso viene convertito in un singolo argomento del tipo "chiave=valore" con il nome della chiave pari a `short-text`.

Ovviamente i segni di uguaglianza `=` all'interno di ambienti matematici, restano segni di uguaglianza e non vengono interpretati come nell'esempio precedente.

Se un segno `=` è presente come segno testuale in un argomento facoltativo, esso va nascosto racchiudendolo fra graffe; per esempio:

```
\caption[{Uso del segno =}]{Uso del segno = negli argomenti facoltativi}
```

Le sottocodifiche di `TS1` sono state introdotte, perché i font diversi da quelli standard gestiti dal LaTeX Team (Computer/Latin Modern) possono non disporre della serie completa dei glifi contenuti nella versione originale. Perciò le sottocodifiche, ovvero i sottoinsiemi vengono definiti nel file `.fd` mediante il comando

`\DeclareEncodingSubset`. Chi crea o gestisce font particolari diversi da quelli standard è quindi tenuto ad aggiungere ai loro file `TS1`⟨*famiglia*⟩`.fd` questo comando, altrimenti viene usata solo l'impostazione di default con risultati tipografici verosimilmente mancanti di diversi glifi. Per maggiori dettagli ci si può riferire al paragrafo 7 del documento che si può leggere con il comando `texdoc fntguide`.

I comandi `\MakeLowercase`, `\MakeUppercase` e `\MakeTitlecase` riconoscono la lingua locale se si usa *babel* e agiscono conformemente; se si devono usare per modificare i font di una ⟨*stringa*⟩ in una lingua diversa da quella corrente, essi accettano un argomento facoltativo, per esempio:

`\MakeUppercase[lang =` ⟨*codice della lingua*⟩`]{`⟨*strings*⟩`}`

dove il ⟨*codice della lingua*⟩ è una sequenza di due o tre lettere conformi al descrittore BCP-47. Una bella tabella che contiene questi codici si trova nel sito `https://en.wikipedia.org/wiki/IETF_language_tag`. Restando nel campo del sistema TeX, la tabella 3 della documentazione di *polyglossia*, riporta tutti i codici BCP 47 che interessano quel pacchetto per la gestione delle lingue (vedi `texdoc polyglossia`).

30.10 La lettera 37 del giugno 2023

È ora disponibile il comando `\IfFileAtLeastTF` che si aggiunge ai comandi `\IfClassAtLeastTF` e `\IfPackageAtLeastTF`. Tutti e tre servono per decidere se fare un non fare quanto descritto nei due ultimi argomenti a seconda che la data del file, della classe o del pacchetto sia posteriore o uguale a quella specificata nel penultimo argomento. Le sintassi sono le seguenti

`\IfClassAtLeastTF{`⟨*classe*⟩`}{`⟨*data*⟩`}{`⟨*vero*⟩`}{`⟨*falso*⟩`}`
`\IfPackageAtLeastTF{`⟨*pacchetto*⟩`}{`⟨*data*⟩`}{`⟨*vero*⟩`}{`⟨*falso*⟩`}`
`\IfFileAtLeastTF{`⟨*file con estensione*⟩`}{`⟨*data*⟩`}{`⟨*vero*⟩`}{`⟨*falso*⟩`}`

La ⟨*data*⟩ va espressa in uno dei formati prescritti dalle norme ISO: *aaaa/mm/gg* oppure *aaaa-mm-gg*. Il nome della ⟨*classe*⟩ o del ⟨*pacchetto*⟩ o del ⟨*file con estensione*⟩ deve essere scritto esattamente come il nome usato dal sistema TeX. Sono stati aggiunti altri tre comandi che ovviano alle limitazioni dei primi due, quasi sempre solo in lettere minuscole. Ovviamente ⟨*vero*⟩ e ⟨*falso*⟩ sono i brani di codice da eseguire se il test è vero oppure se è falso.

I comandi `\lccode` e `\uccode` sono un po' rudimentali per passare dai caratteri minuscoli a quelli maisucoli e non funzionano correttamente con i font Unicode. Ora sono disponibili tre nuovi comandi: `\DeclareLowercaseMapping`, `\DeclareTitlecaseMapping` e `\DeclareUppercaseMapping` per gestire conversioni difficili. Per esempio il comando

`\DeclareUppercaseMapping{"01F0}{\v{J}}`

permette di associare al segno J̌ (indicato col codice Unicode, ora accettato in input anche per pdfLaTeX, producendo il segno J̌, che non è disponibile nei font Type1, i soli accessibili da pdfLaTeX. Maggiori dettagli nella guida `usrguide`.

Il comando `\samepage` è stato migliorato; esso si basa sul valore assegnato a `\predisplaypenalty`, che il precedente `\samepage` impostava in modo che fosse operativo subito prima di ogni ambiente per oggetti mobili. Con la modifica apportata il valore della penalità può essere impostata prima degli oggetti flottanti che devono apparire nella stessa pagina.

Non tutti gli utenti sanno che ogni comando `\label` scrive nel file `.aux` due informazioni; ma il pacchetto *hyperref* ridefinisce questo comportamento in modo da disporre di 5 informazioni distinte. Ora anche i programmi di composizione del sistema TeX usano 5 informazioni anche se *hyperref* non viene usato. I dettagli vanno controllati nella lettera 37, ma è molto interessante ciò che i può fare con queste ulteriori tre informazioni.

30.11 La lettera 38 del novembre 2023

Le novità descritte nella lettera 38 che possono interessare i lettori sono le seguenti.

È stata migliorata la gestione delle 5 informazioni sulle proprietà descritte dal comando `\label` e che sono sistematicamente usate dal pacchetto *hyperref*, ma che, come descritto nella lettera precedente, possono essere usate anche dall'utente con gli appositi comandi predisposti per questo scopo.

È stato aggiunto il comando `\IfExplAtLeastTF` per controllare la data della versione del linguaggio L3 introdotta nel nucleo di LaTeX3. Questa data è diversa da quella del nucleo di LaTeX e richiede un comando apposito. Si vedano i dettagli nella lettera 38.

Sono stati modificati i comandi `\verb` e `\verb*`, nonché gli ambienti *verbatim* e *verbatim** che ora riconoscono i comandi di tabulazione in modo che non siano trattati come un singolo spazio (cosa che succede normalmente nel testo normale) ma vengano riconosciuti come tali e quindi siano inseriti tanti spazi visibili quanto basta per gli incolonnamenti; questo facilita la lettura dei tratti scritti in modo verbatim.

È stato corretto un baco nella gestione delle colonne `p` dell'ambiente *longtable*; questo ambiente avrebbe dovuto seguire le modifiche apportate all'ambiente array, ma un cambiamento in quest'ultimo pacchetto eseguito nel 1992 era stato dimenticato; finalmente è stato riportato anche in *longtable*.

30.12 Le lettere 39 e 40 del giugno 2024

La lettera 40 aggiunge solo qualche dettaglio alla lettera 39. Questa invece è quasi tutta dedicata alle operazioni di upgrade del nucleo di LaTeX al fine di estendere l'operazione di tagging a tutti gli ambienti. In particolare la nuova possibilità di

produrre file archiviabili anche con la sottoestensione 'a', che richiede, appunto, la presenza del tagging. Ora, inoltre è possibile creare file archiviabili con la sottoestensione 'b' direttamente senza bisogno di ricorrere al pacchetto *pdfx*.

Bisogna leggere i numerosissimi dettagli nella documentazione originale, perché è troppo lungo e dettagliato da descrivere qui.

Più interessante per l'utente normale è l'estensione dei test `\IfClassAtLeastTF`, `\IfClassLoadedTF`, `\IfClassLoadedWithOptionsTF`, `\IfFormatAtLeastTF`, `\IfPackageAtLeastTF`, `\IfPackageLoadedTF`, `\IfPackageLoadedWithOptionsTF` sono stati arricchiti delle terminazioni `T` e `F`, cosicché ognuno di quei test ha ora le tre forme, per esempio `\IfClassAtLestaTF`, `\IfClassAtLeastT`, `\IfClassAtLeastF`, di modo che se interessa solo l'azione da compiere se il test è vero, non occorre specificare un codice nullo o `\relax`; e viceversa se interessa il codice da eseguire solo se il test è falso.

Quei comandi a doppio esito erano stati introdotti nel 2020; nel 2023 erano stati aggiunti il test a doppio esito `\IfFileAtLestTF` e `\IfFileLoadedTF`, `\IfLabelExistsTF`, `\IfPropertyExistsTF`, e anche a questi sono stati aggiunti i test a singolo esisto come descritto sopra. Bisogna usare una cautela: i test, come per esempio `\IfFile...`, funzionano solo se il file specificato è dotato della riga `\ProvidesFile` senza la quale il test non ha significato.

Di solito le classi e i pacchetti recenti sono dotati di questa riga, mentre gli altri file meno recenti non erano obbligati ad essere dotati di questa riga. Chi scrive ritiene che la riga formata da `\ProvidesFile` con i suoi argomenti completi di numero di versione e di data siano utilissimi come promemoria per gli autori al fine di poter aggiornare consapevolmente tali file. Perfino il master file di questa guida contiene la riga suddetta, anche se il main file è un normale file `.tex`.

Si ricorda che i file di classe e i pacchetti possono essere caricati solo mediante i comandi `\documentclass`, `\usepackage` e `\RequirePackageWithOptions`, solo nel preambolo o nei pacchetti caricati nel preambolo. Prima LaTeX non verificava questo fatto e se un pacchetto fosse caricato all'interno di un gruppo o un ambiente nel corpo di un documento potevano succedere cose strane. Ora viene eseguita questa verifica e nel caso viene emesso un messaggio d'errore.

Il pacchetto *fontenc* emetteva un messaggio d'avviso se durante l'esecuzione non trovava li file identificativo di una specifica codifica. Ora è stato esteso alla verifica del fatto che font con quella codifica non fanno parte di una installazione, perché evidentemente non è completa.

Ulteriori verifiche sono state implementate per controllare se il nome di un contatore LaTeX può collidere con altri comandi già esistenti; per esempio se un utente definisse `\newcounter{index}`, che oltre al nome definisce il modo di stamparne il contenuto mediante il comando `\theindex`, ma questo collidereb-be col nome dell'ambiente *theindex* che serve per stampare le voci dell'indice analitico.

⋆ ⋆ ⋆

Nei futuri aggiornamenti di questa guida si cercherà di inserire tempestivamente le informazioni che saranno comunicate con le *news letters* successive all'ultima descritta in questo capitolo.

Bibliografia

[1] LaTeX3 Project Team. "LaTeX 2_ε font selection". PDF document, 11 2024. Leggibile con `texdoc fntguide`.

[2] Artech House. "Style Manual". Artech House, Boston – London.

[3] A.V. *The Chicago Manual of Style*. The University of Chicago Press, 14 ed., 1994.

[4] A.V. "Una mica tanto breve introduzione a LaTeX 2_ε". PDF document, 10 2019. In `$TEXMF/doc/latex/lshort-italian/itlshort.pdf`. Traduzione a cura di C. Beccari, G. Pignalberi, M. Sacchetto, T. Gordini, G. Ruocco, G. Milanese.

[5] Claudio Beccari. "La classe TOPtesi". PDF document, 11 2020. In `$TEXMF/doc/latex/toptesi/toptesi.pdf`, `$TEXMF/doc/latex/toptesi/toptesi-it-xetex.pdf`.

[6] Claudio Beccari. *Il LaTeX Refernece Manual commentato*. GuIt, `http://www.guitex.org/home/images/doc/GuideGuIT/`, 2024.

[7] Claudio Beccari. "Regole e consigli per comporre la matematica delle scienze sperimentali". PDF document, 2024. Scaricabile da `http://www.guitex.org/home/images/doc/GuideGuIT/ComporreMatematica.pdf`.

[8] Gian Luigi Beccaria. *Il mare in un imbuto*. Giulio Einaudi editore, Torino, 2010.

[9] Francesco Biccari. "Documentation of the LaTeX class unifith.cls". PDF document, 8 2021. Leggibile con `texdoc unifith`.

[10] Francesco Biccari. "Documentation of the LaTeX class sapthesis.cls". PDF document, 3 2022. In `$TEXMF/doc/latex/sapthesis/sapthesis-doc.pdf`.

[11] Robert Bringhurst. *The elements of typographic style*. Hartley & Marks, Vancouver, BC, 2004.

[12] Christophe Caignaert. "A story of *kpfonts*". In: *TGGboat*, vol. 31, no. 3, (2010), pp. 161–174.

[13] CEI 24-1. *Unità di misura e simboli letterali da usare in elettrotecnica*. Comitato Elettrotecnico Italiano, Milano, 1986.

[14] CEI-S-646. *Dizionario della strumentazione nucleare. Primo complemento al fascicolo S-447*. Comitato Elettrotecnico Italiano, Milano, 1983.

[15] Gustavo Cevolani. "Libretti in LaTeX". In: *ArsTEXnica*, vol. 1, no. 2.

[16] Gustavo Cevolani. "Norme tipografiche". In: *ArsTEXnica*, vol. 1, no. 1.

[17] CNR-UNI 10003. *Sistema internazionale di unità (SI)*. Ente Italiano di Unificazione, Milano, 1984.

[18] Massimiliano Dominici. "Utilizzo di caratteri TrueType con LaTeX – Un esempio pratico: i *FellTypes*". In: *ArsTEXnica*, vol. 2, no. 4, (2007), pp. 88–102.

[19] Massimiliano Dominici. "Una panoramica su Pandoc". In: *ArsTEXnica*, vol. 8, no. 15, (2013), pp. 31–38.

[20] Michael Downes. *Short math guide*. American Mathematical Society, Providence, Rhode Island, 1.09 ed., 2002. In `ftp://ftp.ams.org/pub/tex/doc/amsmath/short-math-guide.pdf`.

[21] Olaf Drümmer, Alexandra Oettler, Dietrich von Seggern. *PDF/A in a Nutshell. Long Term Archiving with PDF*. Callas Software gmbh, 2008. In: `http://www.pdfa.org/doku.php?id=pdfa:en:pdfa_in_a_nutshell`.

[22] Elsevier. "Preparing articles with LaTeX — Instructions to authors for preparing compuscripts". PDF document. In `instraut.pdf`.

[23] Ente Italiano di Unificazione. *Grandezze e unità di misura per la fisica e le scienze sperimentali*. Unificazione Italiana, Milano, 2020. Le norme ISO 80000 sono divise in una quindicina di fascicoli aggiornati in anni diversi, dal 2013 al 2024. Sono acquistabili anche in rete dal sito `store.uni.com/catalogo`.

[24] Giorgio Fioravanti. *Il manuale del grafico — Guida alla progettazione grafica e all'impaginazione del prodotto editoriale*. Zanichelli, Bologna, 1987.

[25] Peter Flynn. *Formatting information — A beginner's introduction to typesetting with LaTeX*. In `CTAN/tex-archive/info/beginlatex/beginlatex-3.6.pdf`.

[26] Michel Goossens. *The XeTeX Companion – TeX meets OpenType and Unicode*. LaTeX Team, gennaio 2011. In `http://xml.web.cern.ch/XML/lgc2/xetexmain.pdf`.

[27] Enrico Gregorio. "Installare TeX Live 2010 su Ubuntu". In: *ArsTeXnica*, vol. ottobre, no. 10, (2010), pp. 7–13.

[28] Enrico Gregorio. "L'arte esoterica di scrivere in cirillico con LaTeX". In: *ArsTeXnica*, vol. aprile, no. 9, (2010), pp. 57–73.

[29] Enrico Gregorio. "Come comporre un frontespizio e vivere felici". PDF document, 9 2011. In `$TEXMF/doc/latex/frontespizio/frontespizio.pdf`.

[30] Enrico Gregorio. "Installing TeX Live 2010 on Ubuntu". In: *TUGboat*, vol. 32, no. 1, (2011), pp. 56–61.

[31] Enrico Gregorio. *Introduzione a XeLaTeX*, gennaio 2011. In `http://guitex.org/home/images/doc/`.

[32] ISO 690. *Information and documentation – Guidelines for bibliographic references and citations to information resources*. International Organization for Standardization, Ginevra, 2010.

[33] ISO Handbook 2. *Units of measurement*. International Organization for Standardization, Ginevra, 1982.

[34] Kluver. "User manual for `kluwer.cls`". In `$TEXMF/doc/latex/kluwer/usrman.dvi`.

[35] Donald E. Knuth. *The TeXbook*. Addison Wesley, Reading, Mass., 16 ed., 1996.

[36] Donald E. Knuth. *Computers & typesetting*. Addison Wesley, Reading, Mass., 2001. Millenium edition.

[37] Helmut Kopka, Patrick W. Daly. *Guide to LaTeX*. Addison Wesley, Reading, Mass., 4 ed., 2004.

[38] Steven G. Krantz. *A primer in mathematical writing*. American Mathematical Society, Providence, Rhode Island, 2 ed., 1998.

[39] Leslie Lamport. *A document preparation system — LaTeX — User's guide and reference manual*. Addison Wesley, Reading, Mass., 2 ed., 1994.

[40] Jerónimo Leal, Gianluca Pignalberi. *Edizioni Critiche – Guida alla composizione con il proprio computer*. TeXnologie. Edizioni CompoMat, Configni (RI), 2012.

[41] Roberto Lesina. *Il nuovo manuale di stile*. Zanichelli, Bologna, 2 ed., 1994.

[42] André Miede. "A classic thesis style". PDF document, 12 2018. In `$TEXMF/doc/latex/classicthesis/classicthesis.pdf`.

[43] Michael Mitchel, Susan Wightman. *Book typography — A designer's manual*. Libanus Press, Marlborough, Wiltshire UK, 2005.

[44] Frank Mittelbach, Michel Goosens, *et al. The LaTeX companion*. Addison Wesley, Reading, Mass., 2 ed., 2023.

[45] Frank Mittelbach, Michel Goosens, *et al. The LaTeX graphics companion*. Addison Wesley, Reading, Mass., 2 ed., 2023.

[46] Bice Mortara Garavelli. *Prontuario di punteggiatura*. Editori Laterza, Bari, 11 ed., 2008.

[47] Bice Mortara Garavelli (cur.). *Storia della punteggiatura in Europa*. Editori Laterza, Bari, 2008.

[48] Imprimerie Nationale (cur.). *Règles typographiques*. Imprimerie Nationale, Parigi, 2006. ISBN 2-7433-0482-0.

[49] David E. Newell, Eite Tiesinga. "The International System of Units", 2019. In `http://physics.nist.gov/cuu/pdf/sp330.pdf`.

[50] Scott Pakin. "The comprehensive LaTeX symbol list". PDF document, 2024. In `$TEXMF/doc/latex/comprehensive/symbols-a4.pdf`.

[51] Lorenzo Pantieri. "Tesi Classica". PDF document, 5 2012. In `http://www.lorenzopantieri.net/LaTeX_files/TesiClassica.zip`.

[52] Lorenzo Pantieri. "Tesi Moderna". PDF document, 5 2012. In `http://www.lorenzopantieri.net/LaTeX_files/TesiModerna.zip`.

[53] Lorenzo Pantieri. "LaTeX per l'impaziente – Un'introduzione all'Arte di scrivere con LaTeX". PDF document, 2017. In `http://www.lorenzopantieri.net/LaTeX_files/LaTeXimpaziente.pdf`.

[54] Lorenzo pantieri. "LaTeXpedia". PDF document, 2019. Scaricabile da `http://www.lorenzopantieri.net/LaTeX_files/LaTeXpedia.pdf`.

[55] Lorenzo Pantieri, Tommaso Gordini. "L'arte di scrivere con LaTeX". PDF document, 2019. In `http://www.lorenzopantieri.net/LaTeX_files/ArteLaTeX.pdf`.

[56] Will Robertson. "The XeTeX reference guide". PDF document, gennaio 2024. In `$TEXMF/doc/xetex/xetexref/XeTeX-reference.pdf`.

[57] Francesca Serafini. *Questo è il punto – Istruzioni per l'uso della punteggiatura*. Editori Laterza, Bari, 2012.

[58] Fabrizio Serra. *Regole editoriali, tipografiche & redazionali*. Istituti Editoriali e Poligrafici Internazionali, Pisa – Roma, 2004. Prefazione di Martino Mardersteig, postfazione di Alessandro Olschki, appendice di Jan Tschichold.

[59] N. Setzer. "The `cool` package". PDF document. In `$TEXMF/doc/latex/cool/cool.pdf`.

[60] Societa Italiana di Fisica e Il Nuovo Cimento, Bologna. *Guida per gli autori*, 1988.

[61] Ellen Swanson, Arlene O'Sean, Antoinette Schleyer. *Mathematics into type*. American Mathematical Society, Providence, Rhode Island, 2 ed., 1999.

[62] Till Tantau. "Ti*k*Z and pgf — Manual for version 3.1.10". PDF document, 2013. In `texlive/2024/texmf-dist/doc/generic/pdf/version-for-pdftex/en/pgfmanual.pdf`.

[63] Ambler Thompson, Barry n. Taylor. "Guide for the use of the International System of units", 2008. In `http://physics.nist.gov/cuu/pdf/sp811.pdf`.

[64] Edward R. Tufte. *The visual display of quantitative information*. Grapphics Press, Cheshire, Connecticut, 2 ed., 2007.

[65] UNI 2949. *Diagrammi e cartogrammi, Regole generali per l'elaborazione*. Ente Italiano di Unificazione, Milano, 1982.

[66] UNI 6015. *Segnaccento obbligatorio nell'ortografia della lingua italiana*. Ente Italiano di Unificazione, Milano, 1967.

[67] UNI 6461. *Divisione delle parole in fin di linea*. Ente Italiano di Unificazione, Milano, 1969.

[68] UNI 7090. *Metodo di scrittura numerica delle date*. Ente Italiano di Unificazione, Milano, 1973.

[69] UNI-ISO 5966. *Presentazione dei rapporti scientifici e tecnici*. Ente Italiano di Unificazione, Milano, 1989.

[70] Ivan Valbusa. "User's guide to suftesi". PDF document, 2023. In `$TEXMF/doc/latex/suftesi/suftesi.pdf`.

[71] Filippo Vomiero. *Utilizzare i software per la gestione della bibliografia con LaTeX*. GuIt, 7 2019. Scaricabile da `https://www.guitex.org/`.

[72] Herbert Voss. “Math mode”. PDF document, 2010. In `$TEXMF/doc/latex/mathmode/`.

[73] R.C. Weast, M.J. Astle, W.H. Beyer (cur.). *CRC handbook of chemistry and physics*, CRC Press, Boca Raton, Florida, cap. “Symbols, units and nomenclature in physics”, pp. F259–F293. 65 ed., 1984.

[74] Peter Wilson. *A few notes on book design*. The Herries Press, Normandy Park, WA, 2018.

[75] Peter Wilson, Lars Madsen. “The memoir class for configurable typesetting — User guide”. PDF document, 2024. In `\protect\T1\textdollarTEXMF/doc/latex/memoir/memman.pdf`.

Indice analitico

Simboli

C

D

E

F

G

H

I

O

Q

R

S

T

Indice degli ambienti

Indice delle classi

Indice dei file

Estensioni

Indice dei pacchetti

Indice dei programmi e delle distribuzioni

N

O

P

Q

Y

Z