Numero 23
Aprile 2017

ArsTEXnica

Rivista italiana di TEX e LATEX

GUIT

http://www.guitex.org/arstexnica/

GUIT – Gruppo Utilizzatori Italiani di TEX

ArsTEXnica è la pubblicazione ufficiale del GUIT

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

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

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

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

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

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

arstexnica@guitex.org.

Nota sul Copyright

Associarsi a GUIT

Fornire il tuo contributo a quest'iniziativa come membro, e non solo come semplice utente, è un presupposto fondamentale per aiutare la diffusione di TEX e LATEX anche nel nostro paese. L'adesione al Gruppo prevede una quota di iscrizione annuale diversificata: 30,00 € soci ordinari, 20,00 (12,00) € studenti (junior), 75,00 € Enti e Istituzioni.

Indirizzi

Gruppo Utilizzatori Italiani di TEX
c/o Università degli Studi di Napoli Federico II
Dipartimento di Ingegneria Industriale
Via Claudio 21
80125 Napoli – Italia
`http://www.guitex.org`
guit@guitex.org

Redazione ArsTEXnica:
`http://www.guitex.org/arstexnica/`
arstexnica@guitex.org

Codice ISSN 1828-2369

Stampata in Italia
Napoli: 15 Aprile 2017

ArsTEXnica

Rivista italiana di TEX e LATEX

Numero 23, Aprile 2017

Claudio Beccari
Editoriale 3
Claudio Beccari
Comporre in griglia 5
Roberto Giacomelli
A Database Experiment With LuajitLATEX 12
Werner Lemberg
Storia della notazione musicale. Una ricognizione per immagini 35
Gianluca Pignalberi
Riviste di aspetto complesso e composte a griglia con LATEX 55
Hans Hagen, Luigi Scarso
Foreign Function Interface in LuaTEX 70

Gruppo Utilizzatori Italiani di TEX

Editoriale

Claudio Beccari

Il Meeting di Brescia si è svolto in modo particolare; per serie cause familiari il Presidente non ha potuto intervenire, cosicché non si è potuta svolgere l'assemblea dei soci per procedere al rinnovo del Consiglio Direttivo della nostra associazione. Ora, febbraio 2017, le operazioni di rinnovo sono in corso mentre sto scrivendo questo editoriale; faccio gli auguri ai nuovi eletti e al nuovo Direttivo che avrà il compito di condurre la nostra associazione per i prossimi anni.

In questo numero di ArsTEXnica della primavera 2017 ci troviamo di fronte ad alcuni importanti contributi e sono convinto che i lettori li troveranno molto interessanti.

Roberto Giacomelli usa il sistema TEX per produrre tutti i documenti inerenti alla sua professione di ingegnere. La quantità e varietà di documenti che deve produrre è molto grande e ha già sviluppato diversi metodi per automatizzare buona parte dell'integrazione delle informazioni sparse in diversi file per produrre singoli documenti completi. Da un po' di tempo si è orientato verso l'uso di LuaLATEX e del linguaggio di scripting Lua, per svolgere compiti difficilmente affrontabili con le funzionalità native degli altri interpreti del sistema TEX o del mark-up di LATEX. Nel suo articolo di questo numero di ArsTEXnica ci presenta i risultati che ha ottenuto usando LuajitLATEX per gestire database da usare nei documenti da comporre; la stringa misteriosa "jit" contenuta nel nome del programma è l'acronimo di "Just In Time" e lascia intendere che questa nuova versione sia in grado di importare in tempo reale moduli di codice presenti in librerie scritte in codice macchina o in linguaggi diversi da quelli normalmente usati dai programmi di composizione del sistema TEX. I suoi risultati sono molto interessanti e possono indicare la via anche per altri utenti che devono integrare contenuti informativi diversi in un unico documento.

Werner Lemberg ci ha gentilmente concesso il permesso di tradurre e pubblicare il suo articolo sulla storia della scrittura musicale apparso sul numero 3 di *TUGboat* 2016. Grazie a Tommaso Gordini, noto appassionato di LATEX e anche musicista, per la sua traduzione in italiano. Lemberg ha esposto la storia della notazione musicale dalle origini fino ai tempi nostri: dall'epoca mesopotamica fino alla composizione tramite il calcolatore. L'articolo è ricchissimo di immagini, una più interessante dell'altra, validissime sia per la persona colta in generale sia per chi conosce la musica. Quindici pagine di immagini a fronte di quattro pagine di testo ha richiesto una impaginazione particolare che, speriamo, non sia sgradita ai lettori.

Ringraziamo Werner Lemberg che ci ha concesso di tradurre il suo articolo; speriamo che non se ne abbia a male se lo abbiamo impaginato in modo diverso da come l'articolo è apparso su *TUGboat*.

Gianluca Pignalberi ci descrive le sue esperienze per la realizzazione di una rivista dall'impaginazione complessa e con il testo composto in griglia. Egli ha una lunga esperienza con l'impaginazione di riviste; e l'impaginazione in griglia è un problema che ha affrontato nell'arco degli ultimi 10 anni con risultati via via più soddisfacenti. Quanto ha ottenuto con i suoi ultimi progressi in questo campo è decisamente notevole anche se ha dovuto accettare alcuni compromessi, inevitabili quando si compongono stampati che contengono anche testo composto con font di diversi corpi.

Luigi Scarso e Hans Hagen si occupano da tempo di sviluppare l'integrazione di LuaTEX con vari tipi di linguaggi di scripting e di librerie esterne in modo da far svolgere a questo interprete i compiti tipici dei programmi di composizione del sistema TEX in modo più efficiente ed efficace. In questo articolo essi presentano gli esperimenti che hanno condotto per stabilire le migliori procedure adatte per font OpenType che contengano non solo i caratteri latini ma anche insiemi di caratteri "esotici", complessi, e ricchi di legature, come per esempio i caratteri arabi; d'altra parte font latini OpenType come gli EBGaramond richiedono un trattamento speciale a causa della moltitudine di varianti che essi consentono di usare. In sostanza il loro studio riguarda l'integrazione di una funzione esterna (*foreign function integration*, 'ffi') per svolgere non solo i compiti riferiti ai font, ma qualunque funzione esterna che possa essere utile per il processo di composizione. L'argomento trattato è ancora *work in progress*, ma si intravedono già gli sviluppi che si potranno ottenere e che poi saranno disponibili per tutti gli utenti.

L'articolo di Claudio Beccari rappresenta un piccolo approccio molto empirico alla composizione in griglia, che Pignalberi nel suo articolo affronta in modo più professionale. Il mio concetto è che non si può avere tutto composto in griglia, specialmente gli inserti fuori testo come le formule matematiche, le figure, le tabelle, le citazioni in display e in corpo minore di quello normale e altri brani di testo particolari. Come fare allora? Si agisce a mano per aggiustare la posizione sulla griglia; ma per farlo è necessario "vedere" la griglia; ecco allora una

piccola macro che permette di disegnare la griglia sotto il testo da comporre, come fosse l'insieme di righe di una pagina di un quaderno per le scuole elementari; così si vede dove è necessario inserire piccole spaziature verticali positive o negative per riportare il testo normale sulla griglia, ma lasciando gli elementi fuori testo composti senza tenere conto della griglia. Il compromesso c'è lo stesso, ma non ci sono automatismi che evitino l'aggiustamento fine eseguito a mano.

Ringrazio moltissimo i membri del comitato di redazione, il consiglio scientifico che ha esaminato e valutato gli articoli qui pubblicati, i revisori editoriali e i correttori che sono intervenuti sugli articoli in italiano e in inglese per migliorarne il testo. Grazie agli autori e grazie allo staff di ArsTEXnica, la rivista si mantiene all'alto livello che è sotto gli occhi dei lettori.

▷ Claudio Beccari
Professore emerito
Politecnico di Torino
`claudio dot beccari at gmail dot com`

Comporre in griglia

Claudio Beccari

Sommario

Nei manuali tipografici si raccomanda sempre di comporre appoggiando le righe del testo su una griglia ideale identica sul recto e sul verso di ogni singola pagina. Il motivo sarebbe quello di non lasciar trasparire il testo di una faccia della pagina se non nascondendone le righe dietro le righe dell'altra faccia.

È possibile farlo anche componendo con i programmi del sistema TEX?

Abstract

Handbooks on typography always recommend to lay the written lines on both sides of the same page on an ideal grid identical for the recto and verso sides of each page. The motivation should be to avoid that the lines on one side of the page get visible while reading the other side of the same page.

Is it possible with the means of the various typesetting programs of the TEX system?

1 Introduzione

Fin dall'inizio Knuth si è preoccupato di creare un sistema di tipocomposizione elettronica in grado di usare un scartamento costante fra le righe delle due pagine stampate sul recto e sul verso della stessa pagina; non solo ma anche componendo a due colonne è bene che le righe della prima siano allineate con quelle della seconda colonna; lo stesso vale, sia pure, forse, con minor peso, fra le righe di due pagine affacciate.

Se tutto il testo fosse composto in questo modo, si direbbe che la composizione è "in griglia".

Ma è vero che questo avviene spontaneamente quando si usano i programmi di tipocomposizione del sistema TEX?

La risposta è negativa, a meno che non si aggiustino le cose a mano, ma il "no" si riferisce soprattutto all'avverbio "spontaneamente".

La prima volta che si è visto trattare questo argomento è stato in un articolo di Frank Mittelbach (Mittelbach, 2000) il cui titolo faceva riferimento agli oggetti mobili, ma se ne trattava la posizione anche in relazione al fatto di mantenere il testo collocato sulla griglia nonostante l'ingombro di tali oggetti.

Diversi anni fa Gianluca Pignalberi ha pubblicato un articolo, (Pignalberi, 2005), nel quale descrive come ha realizzato la composizione di una rivista con LATEX, dove ha lavorato in colonne strette, con molti riquadri, e altre cose che normalmente si trovano nelle riviste stampate in rotocalco. Evidentemente si è anche preoccupato della composizione in griglia, proprio perché le colonne affiancate metterebbero in risalto la mancanza delle composizione in griglia.

Durante quel meeting del 2005 ne ha evidentemente parlato con Kaveh Bazargan che evidentemente doveva fronteggiare gli stessi problemi; l'anno successivo, infatti, Kaveh Bazaragan pubblicava su ArsTEXnica un articolo sull'argomento, (Bazargan e Radakrishnan, CV, 2006).

Non molto tempo dopo, le distribuzioni dei sistemi TeX si sono arricchite di un nuovo pacchetto, (River Valley Technologies, 2009), dove era stato realizzato quanto Kaveh Bazargan aveva annunciato con il suo articolo su ArsTEXnica.

In sostanza l'articolo Bazargan e Radakrishnan, CV (2006) preannunciava quello che il pacchetto grid aveva realizzato, (River Valley Technologies, 2009). In parole povere si trattava di ridefinire tutti i comandi, generalmente per la composizione di ambienti che producono materiale fuori testo, dalle figure alla matematica in display, solo per citarne alcuni. Il manuale di grid elenca alcuni casi che non sono stati presi in considerazione; ma l'idea che se ne ricava è che il problema è decisamente ancora molto aperto.

In questo articolo si cerca di spiegare la difficoltà intrinseca nella composizione in griglia e quali accorgimenti si possono prendere per superare alcuni problemi. Ritengo che esistano sufficienti mezzi TEXnici per poter eseguire la rifinitura finale di un documento da comporre in griglia agendo a mano e lasciando che l'utente possa scegliere criticamente quale espediente usare per rimettere il testo in griglia, quando per qualche motivo la composizione in griglia sia disturbata.

2 Che cosa impedisce di comporre in griglia

Forse elencherò delle cose ovvie, ma se non ci si pensa prima, si rischia di ritrovarsi con la griglia disturbata. Nello stesso tempo è meglio sapere che cosa si può e che cosa non si può ottenere.

La composizione in griglia può andare bene forse in un documento di solo testo come un romanzo. Questo è molto semplice; il romanzo non è strutturato tipograficamente; se è diviso in capitoli e se lo stile della pagina dei capitoli è costruita correttamente, il testo che segue l'intestazione può essere composto in griglia, perché è un testo composto

con lo stesso font e senza cambiamenti di corpo. Se il romanzo non contiene figure, la regolarità delle righe di testo rimane integra e il tutto viene composto in griglia.

Se il documento è di genere letterario ma contiene delle figure i problemi si pongono; il pacchetto grid dovrebbe essere in grado di risolverli. Ma sono proprio le figure fuori testo che, con le loro dimensioni scorrelate da quelle del testo non possono rispettare la griglia. Lo stesso vale per un testo contenente tabelle e altri oggetti mobili.

Ma un testo tipograficamente strutturato ha moltissime cose fuori testo oltre gli oggetti mobili e le eventuali espressioni matematiche in display. Per esempio tutti i titoli; per esempio il contenuto di brani dentro gli ambienti quote e quotation; brani composti in stile epigrafico, o in bandiera a sinistra o a destra; le liste; i cambiamenti di corpo. La matematica pone i soliti problemi, non solo perché si tratta di oggetti decisamente più alti di una riga, ma anche perché le grosse strutture matematiche potrebbero occupare parecchio spazio verticale in quanto contengono grandi operatori, delimitatori estensibili, frazioni, determinanti, matrici ed altri oggetti già grandi per loro conto.

Ma esistono anche i cambiamenti di corpo dei font; con questi lo scartamento ideale ammonta a circa il 20% in più del corpo; ecco quindi che un testo composto in corpo 12 pt ha uno scartamento di 14,4 pt, ma una citazione in display in corpo 11 pt, ha uno scartamento di 13,2 pt, e le note sono composte in corpo 10 pt con uno scartamento di 12 pt. Ecco quindi che con scartamenti diversi la composizione in griglia ne risulta disturbata. Si può impostare ogni corpo minore in modo che mantenga lo scartamento del corpo normale, ma a questo punto l'equilibrio tra corpo e scartamento ne risulta disturbato.

Se una breve espressione matematica resta in linea col testo si possono avere sorprese inaspettate; è vero che la composizione della matematica in linea col testo avviene in modi diversi rispetto a quella della matematica in display, tuttavia una piccola espressione come $\sqrt{a^2+b^2}$ occupa meno spazio verticale dell'espressione $\sqrt{\sqrt{a^2+b^2}+c^2}$ e la differenza si nota nella diversa interlineatura della riga che contiene la seconda espressione. Anche in questo caso la griglia risulta disturbata.

Consideriamo un brano di testo citato mediante l'ambiente quote senza ricorrere ad un corpo minore; l'ambiente quote non solo compone il suo argomento con una giustezza minore del testo circostante, ma anche distanzia sopra e sotto il testo citato mediante spazi che non hanno nulla a che vedere con lo scartamento; questi spazi si potrebbero aggiustare, ma per mantenere la griglia bisognerebbe usare spazi di valore pari o simile a quello dello scartamento, ma uno spazio così grande risulterebbe inestetico.

Consideriamo una figura con la sua didascalia; questo oggetto di grandi dimensioni viene di solito trattato come un oggetto mobile. In quanto tale, esso viene messo dentro una delle "scatole" che vengono usate in continuazione dal motore di composizione, e la scatola viene deposta in uno "scaffale" ordinato, che informaticamente parlando è una catasta del tipo FIFO: *First In, First Out*; da qui ogni scatola viene prelevata dal motore di impaginazione solo quando su una pagina c'è abbastanza spazio per inserirla. Quando questo spazio risulta disponibile, il motore di impaginazione inserisce nella pagina in costruzione l'oggetto mobile, eventualmente preceduto e/o seguito da spazi adeguati per evitare il contatto, comunque per mettere in evidenza l'oggetto; un po' come descritto per i testi delle citazioni, questi spazi non possono essere troppo grandi, e quindi non hanno relazione con lo scartamento delle righe di testo. Questi spazi sono definiti dalla classe ma, a differenza di altre situazioni, le figure e le tabelle possono essere anche adiacenti verticalmente fra di loro, oltre che al testo circostante, e quindi non si possono accumulare spazi che distanzierebbero troppo quando, appunto, questi spazi vengono accumulati. Non è il solo motivo per il quale la griglia risulta disturbata, perché bisogna tenere conto che la scatola ha un punto di riferimento (generalmente) coincidente con lo spigolo inferiore sinistro della scatola; se questo punto di riferimento venisse posto su una riga della griglia, anche se la scatola avesse un'altezza complessiva pari ad un multiplo intero dello scartamento, il lato superiore della scatola potrebbe toccare la linea di base della riga (bianca) sottostante alla linea di base dell'ultima riga di testo, e quindi lo spazio di distanziamento dovrebbe essere decisamente inferiore allo scartamento; questo è un altro motivo per il quale si disturba la griglia.

Infine le impostazioni della classe o prodotte da diversi pacchetti usano spesso lunghezze elastiche, specialmente quando la classe può comporre i testi per una stampa fronte-retro; generalmente questi spazi elastici presenti, per esempio, all'inizio di ogni capoverso hanno una lunghezza naturale nulla, ma hanno una componente di allungamento non nulla; questo nasce dal fatto che quando si compone fronte-retro è bene che la base degli specchi di stampa delle due pagine affacciate siano alla stessa distanza dal bordo delle pagine; questa è l'impostazione `\flushbottom` predefinita. Ma i motori di composizione del sistema TEX non fanno gli errori madornali che hanno luogo con i normali word processor, per esempio quello di comporre il titolo di un paragrafo in fondo alla pagina per poi cominciare il primo capoverso del paragrafo nella pagina successiva. I programmi di messa in pagina del sistema TEX rimandano il titolo alla pagina successiva, e allungano gli spazi verticali della pagina

in questione sfruttando l'allungabilità degli spazi elastici che normalmente non danno segni della loro presenza. Siccome questi spazi allungati talvolta sono allungati veramente troppo (con la magra consolazione di un messaggio di avvertimento nel file `.log` che quella pagina è "Underfull"), si potrebbe impostare la dichiarazione `\raggedbottom`, ma a questo punto gli specchi di stampa di due pagine affacciate non terminerebbero alla stessa distanza dalla base della pagina; sì, saltuariamente un pagina mozza è accettabile, ma non che tutte le pagine di una documento abbiano altezze diverse.

Ecco, descritti a parole, questi sono i problemi che si presentano quando si vorrebbe comporre in griglia dei testi tipograficamente strutturati e contenenti oggetti di grandi dimensioni. Vi pone rimedio parziale il pacchetto grid ma a spese di spazi verticali troppo grandi; anche questo pacchetto non può fare miracoli.

3 Rimedi ad alcuni difetti della composizione in griglia

Qui si indicheranno alcuni rimedi ai problemi evidenziati, non a tutti, perché alcuni richiedono soluzioni di compromesso.

3.1 Oggetti in linea con il testo

Riprendiamo l'esempio della matematica in linea con il testo $\sqrt{\sqrt{a^2+b^2}+c^2}$; esso è troppo alto; lo stesso problema si potrebbe presentare con un logo, per esempio quello tondo del GUIT,  . È chiaro che rimpicciolire il logo tondo del GUIT fino ad essere alto quanto lo scartamento significa forzare l'esempio, perché un logo reso così piccolo è irriconoscibile, tuttavia rende l'idea di quello che si può fare; basta usare il comando `\smash` e incrociare le dita; se va bene l'oggetto viene sistemato nella riga come un oggetto privo di dimensioni verticali (*to smash*, schiacciare), senza provocare nessuna interlineatura e quindi senza disturbare la griglia: $\sqrt{\sqrt{a^2+b^2}+c^2}$ vale per la formula e  vale per il logo. Per fortuna nell'esempio della formula, la parola della riga sovrastante non ha discendenti, quindi non si ha alcuna interferenza; sempre in questo esempio il logo batte appena contro una lettera della riga precedente, ma in una maniera "accettabile".

In realtà né un caso né l'altro sono accettabili; per avere la composizione in griglia si è dovuta fare una bruttura, tipograficamente inaccettabile. Per la formula, senza usare `\smash`, si potrebbe localmente usare un corpo inferiore, per esempio $\sqrt{\sqrt{a^2+b^2}+c^2}$. Ma per il logo, quel logo, non ci sono soluzioni come ridurlo in scala:  ; illeggibile era e ancor più illeggibile resta.

Diverso sarebbe un disegno eseguito con i mezzi del sistema TEX, per esempio con l'ambiente picture aggiornato a LATEX 2_ε mediante il pacchetto pict2e; il simbolo della *command key*, ⌘, delle macchine Apple può essere disegnato con il codice seguente

```
\newcommand*\commandkey{{\unitlength=0.1ex
\begin{picture}(18,18)(0,-3)
\linethickness{2\unitlength}
\moveto(6,0)\lineto(12,0)
\curveto(16.5,0)(18,-1.5)(18,-3)
\curveto(18,-4.5)(16.5,-6)(15,-6)
\curveto(13,-6)(12,-3)(12,0)\lineto(12,6)
\curveto(12,9.5)(13.5,12)(15,12)
\curveto(16.5,12)(18,10.5)(18,9)
\curveto(18,7.5)(15,6)(12,6)\lineto(6,6)
\curveto(3,6)(0,7.5)(0,9)
\curveto(0,10.5)(1.5,12)(3,12)
\curveto(4.5,12)(6,10.5)(6,6)\lineto(6,0)
\curveto(6,-3)(4.5,-6)(3,-6)
\curveto(1.6,-6)(0,-4.5)(0,-3)
\curveto(0,-1.5)(1.5,0)(6,0)
\strokepath
\end{picture}}}
```

Avendo parametrizzato tutto il disegno alla dimensione di 0,1ex, il disegno ingrandisce e rimpicciolisce al variare del corpo del font in uso, ma anche nel disegno diventa relativamente facile evitare di superare il corpo stesso e a maggior ragione lo scartamento associato a quel corpo.

In generale gli oggetti troppo grandi da inserire in linea con il testo o si riescono a rimpicciolire (al massimo del 10%) oppure non si usano del tutto oppure li si mette in display.

3.2 Una citazione in displyay

Si faccia riferimento alla figura 1 dove sono esplicitamente disegnate le linee di base della griglia e su cui è sovrapposto un po' del testo precedente e un po' del testo successivo ad un breve brano di testo composto con l'ambiente quote.

...l'ammiraglio approvò la manovra e citò la famosa frase latina di Nelson:

> Semper ad maiora, nautae britannici vicerunt!

prommettendo un meritato encomio ai marinai.

FIGURA 1: Griglia: un breve testo con citazione in display

Come si vede il testo riprende dopo la citazione sulla griglia; la citazione di due righe è spaziata dal testo circostante in modo da occuparne esattamente tre righe; è ovvio che anche la citazione potrebbe avere le sue righe sulla griglia se gli spazi verticali di separazione fossero aumentati in modo che complessivamente formino una riga in più. Lo spazio prima e dopo la citazione in display è lo stesso usato per le liste ed è formato dalla somma dei tre parametri `\topsep`, `\parskip` e `\partopsep`. Questi valori sono specificati dalla classe e sono diversi da classe a classe. Il parametro `\partopsep`

viene usato solo quando l'ambiente della citazione è preceduto da una riga vuota, cosicché la citazione inizia un nuovo capoverso; `\parsep` ha un valore positivo solo negli ambienti di tipo "lista" dove le voci possono essere formate da più capoversi senza rientro; `\topsep`, invece, ha quasi sempre un valore positivo in tutte le liste, ma in particolare nella "lista" quote. Siccome non si sa a priori se siano stati impostati valori diversi, bisogna supporre che i tre parametri abbiano valori non nulli, e quindi bisognerebbe determinare dinamicamente, quanto bisogna aggiungere in effetti allo spazio di separazione per avere un valore complessivo pari a quello dello scartamento; occorre aggiungere nella definizione del comando di apertura un calcolo del tipo:

```
\setlength{\topsep}{\dimexpr\baselineskip
   -\topsep -\parsep -\partopsep \relax}
```

Un'operazione del genere dovrebbe essere eseguita per ogni ambiente di tipo "lista", come enumerate, itemize, description; per ogni altro ambiente definito o definibile con le funzionalità di pacchetti come enumitem; per ogni ambiente come il già citato quote e gli ambienti quotation, verse, centering, flushleft, flushright e simili che non sono liste, ma che il mark up LaTeX implementa per mezzo di liste. Bisognerebbe anche che tutti questi ambienti fossero sempre impostati con `\parsep` nullo ma con il rientro dei capoversi non nullo, in modo che anche dentro l'ambiente si mantenga la composizione in griglia.

Come è facile immaginare l'operazione non è per niente facile ed è praticamente impossibile da fare in tutti gli ambienti definiti dai pacchetti più disparati presenti in ogni distribuzione del sistema TeX.

3.3 I brani di testo composti in corpo diverso dal normale

Questo riguarda specialmente le note e le didascalie composte in corpo minore del normale. Il problema è evidentemente insolubile, come si è già spiegato nella sezione precedente: se si mantiene lo stesso scartamento del corpo normale, le righe in corpo minore sono composte male con uno scartamento eccessivo; se si usa lo scartamento ottimale per il corpo del font specifico usato in questi testi, non si riesce a mantenere la composizione in griglia.

Secondo un parere personale il problema è di minore importanza, e sarebbe opportuno mantenere lo scartamento ottimale relativo al font di corpo minore. Infatti, se lo scopo della composizione in griglia è quello di non far trasparire il testo stampato su una faccia di una pagina quando si legge l'altra faccia della stessa pagina, allora bisogna ricordare che le didascalie si trovano dentro la "scatola degli oggetti mobili", e trovano posto dove il motore di impaginazione riesce a collocare tali oggetti; l'importante è che l'oggetto non disturbi

Figura 2: Una variante del logo tondo del GUIT

Figura 2: Griglia: una immagine mobile inserita fra il testo e che disturba la griglia

la composizione in griglia. Un discorso analogo si può fare per le note al piede; in generale le note al piede non compaiono in tutte le pagine e in ogni caso compaiono solo al piede, quindi il disturbo alla griglia influisce solo sulla parte bassa dello specchio di stampa e non su tutta, o quasi tutta la pagina.

3.4 Gli oggetti mobili

Gli oggetti mobili, come le figure e le tabelle, presentano altri problemi; si veda infatti la figura 2 dove, sopra lo sfondo della griglia, è riportata la scatola di una figura.

La figura 2 mostra chiaramente il disturbo apportato alla composizione in griglia da parte delle dimensioni del disegno. Infatti il "testo successivo" è posto sulla griglia. L'ultima riga della didascalia (che in questo esempio è formata da una sola riga) è posta sulla griglia, e fra questi due elementi vi è lo spazio definito dal parametro `\intextsep` specificato dalla classe del documento. Sopra, spaziato dallo spazio `\abovecaptionskip` definito dalla classe, c'è l'immagine introdotta con `\includegraphics` e scalata in modo che abbia la larghezza pari al 70% della giustezza; sopra ancora c'è il "testo precedente" separato dall'immagine mediante lo stesso spazio `\intextsep`.

Per riaggiustare la composizione in griglia, quindi, non si possono modificare gli spazi, ma si deve modificare il fattore di scala dell'immagine; nella figura 3 lo si è fatto trovando sperimentalmente che il coefficiente da applicare alla larghezza non è `0.7`, ma `0.685`.

È chiaro che questo tipo di aggiustamenti si deve fare a mano, ma senza vedere la griglia, come si è

FIGURA 3: Una variante del logo tondo del GUIT

FIGURA 3: Griglia: una immagine mobile inserita fra il testo e che non disturba la griglia

fatto nelle figure 1, 2 e 3 è impossibile controllare se la griglia è disturbata.

3.5 Spazio fra i capoversi

Quando il programma impaginatore è costretto ad allargare lo spazio elastico fra un capoverso e l'altro, perché non ha abbastanza spazio in fondo alla pagina per inserire un grosso oggetto o un titolo con almeno due righe di testo, secondo il parere personale è meglio provare ad aggiustare le cose con uno dei tre metodi seguenti.

1. Modificare il testo dei capoversi, contenuti nella pagina con gli spazi allargati, aggiungendo qualche parola o spostando qualche parola o qualche locuzione in un'altra posizione. Almeno in italiano, che non è così posizionale come l'inglese, questa operazione si può fare con una certa libertà, anche se, a rigore, questa operazione può modificare non tanto il senso, quanto il registro del periodo.

2. Modificare il numero di righe di uno o più capoversi nella pagina problematica usando il poco conosciuto comando primitivo `\looseness`; la sintassi è basilare, perché si tratta di un comando nativo di TEX:

   `\looseness`⟨*numero intero*⟩

   Il comando va inserito all'interno di un capoverso, al massimo come suo ultimo elemento prima della riga vuota; esso rimane in vigore solo per quel capoverso; il ⟨*numero intero*⟩ da mettere così, senza graffe, indica di quante righe stiracchiare il capoverso (se il numero è positivo) o comprimere (se il numero è negativo). In pratica funziona con capoversi relativamente lunghi e se finiscono con un righino contenente solo una parola non troppo lunga (usare allora il numero −1) o una riga a cui manca poco per arrivare alla fine dello spazio orizzontale (usare allora il numero +1). Spesso non si ottengono risultati diversi perché quel comando non è un "ordine" ma un suggerimento del tipo "se ci riesci allunga"/accorcia il capoverso di una riga". Ma altrettanto spesso il risultato è proprio quello che si voleva.

3. Mozzare la pagina purché sia sufficientemente piena, cioè manchino allo specchio di stampa poche righe per essere pieno, ma non abbastanza per inviare il titolo e almeno due righe alla pagina successiva. Il comando `\goodpagebreak` definito così

   ```
   \newcommand\goodpagebreak[1][4]{\par%
   \unless\ifdim
      \dimexpr\pagegoal-\pagetotal
        > #1\baselineskip\newpage\fi\par
   }
   ```

   permette di introdurre il comando `\newpage` solo se la differenza fra il `\pagegoal` e il `\pagetotal` non è superiore al numero di righe specificato con l'argomento facoltativo del comando. Il parametro dimensionale `\pagegoal` assomiglia all'altezza della gabbia di stampa, ma ne differisce via via che il motore di costruzione della pagina aggiunge note in calce; `\pagetotal` è l'altezza della pagina parziale costruita fino al momento di eseguire quel comando. La sintassi del comando è:

   `\goodpagebreak[`⟨*numero*⟩`]`

   Dove ⟨*numero*⟩ è un numero reale, sebbene in pratica si usino numeri interi. Benché la definizione di `\goodpagebreak` inizi e finisca con comandi `\par` che ordinano la fine di un capoverso, il comando deve essere inserito nel file sorgente preceduto e seguito da righe vuote. Normalmente non è necessario specificare l'argomento facoltativo, ma talvolta, specialmente prima di un titolo che potrebbe occupare più di una riga, potrebbe essere necessario usare un numero maggiore del valore preimpostato 4, per esempio 5 o raramente 6.

   Componendo a due colonne, e usando `\goodpagebreak`, il comando `\newpage` tronca la colonna corrente e l'impaginatore comincia la colonna successiva, eventualmente in una nuova pagina.

Come si vede ci sono diverse possibilità ma tocca sempre all'utente valutare opportunamente quale delle soluzioni usare.

3.6 Commenti

Gli esempi mostrati nei paragrafi precedenti mostrano chiaramente i problemi che nascono dall'esigenza di comporre in griglia. LaTeX avrebbe gli strumenti per farlo, ma non in modo automatico; il pacchetto grid fa del suo meglio, anzi fa tutto il possibile, ma non fa tutto il necessario, anche perché le casistiche da aggiustare sono troppe.

Solo l'intelligenza dell'operatore può valutare il tipo di intervento da eseguire caso per caso e può intervenire a mano sulla base delle sue valutazioni.

4 Mostrare la griglia

Un aiuto necessario all'operatore è quello di poter disporre della griglia come immagine di sfondo. Non mancano i pacchetti per inserire le immagini di sfondo, ma siccome la griglia deve essere legata ai parametri geometrici del disegno grafico della pagina, penso che sarebbe meglio definire una procedura che provveda direttamente.

Occorre una variabile booleana per determinare se si vuole o non si vuole la griglia, diciamo `\ifgriglia`; occorre un valore fisso dello scartamento relativo al font normale, e che non venga modificato da nessun cambiamento di corpo; occorre un punto di ancoraggio fisso del disegno della griglia che non dipenda dal corpo in uso, ma solo dal disegno della pagina; possibilmente queste cose non devono dipendere dalla classe usata, nel senso che qualunque sia la classe, lo sfondo della griglia deve risultare nella posizione corretta.

Per fare queste cose il disegno della griglia deve essere parametrizzato alla fine del preambolo e non deve dipendere da eventuali (inopportune) impostazioni di modifica dello scartamento eseguite con il comando `\linespread{`⟨*coefficiente*⟩`}` nel corpo del documento.

Il punto di riferimento può essere scelto come l'angolo inferiore sinistro della testatina; la griglia poi dovrà tenere conto che le righe da tracciare devono prendere in considerazione un offset dovuto sia allo spazio distanziatore della testatina dall'inizio del testo e del valore `\topskip` della prima riga del testo; sarebbe meglio che le righe della griglia siano tracciate in numero pari alle righe dello specchio di stampa; questo numero va calcolato in modo corretto (ovviamente) tenendo conto che i calcoli che i programmi del sistema TeX eseguono sono approssimati alla quinta cifra decimale.

L'ambiente picture è in grado di fare tutto ciò, ma per collocare la griglia in modo corretto bisogna definire le testatine in modo acconcio. Se in un dato documento esse sono impostate con le funzionalità del pacchetto fancyhdr tanto meglio, altrimenti bisogna ridefinire i comandi dello stile delle pagine `\ps@empty`, `\ps@plain`, `\ps@headings` e `\ps@myheadings`; eventualmente anche altri stili definiti dalla classe in uso.

In tutti i tipi di testatina, anche se inizialmente vuota, bisogna aggiungere il comando `\griglia` come primo elemento a sinistra dell'unico campo o del campo a sinistra di ogni testatina; lo si può fare avvalendosi delle funzionalità del pacchetto xpatch se bisogna agire direttamente sugli stili delle pagine, oppure aggiungendo a mano tale comando nelle specificazioni mediante i comandi di fancyhdr. Qui mostro come fare solo per le testatine dello stile `empty`.

Modifica diretta dello stile Lo stile `empty` è definito nel nucleo di LaTeX i questo modo:

```
\def\ps@empty{%
  \let\@mkboth\@gobbletwo
  \let\@oddhead\@empty
  \let\@oddfoot\@empty
  \let\@evenhead\@empty
  \let\@evenfoot\@empty}
```

Bisogna ridefinirlo nel modo seguente[1]:

```
\makeatletter
\renewcommand{\ps@empty}{%
\let\@mkboth\@gobbletwo
\let\@oddfoot\@empty}
\let\@evenfoot\@empty
\def\@oddhead{\griglia\hfil}
\let\@evenhead\@oddhead
\makeatother
```

Modifica tramite fancyhdr Il pacchetto fancyhdr permette di definire nuovi stili di testatine e di ridefinire gli stili predefiniti anche nel nucleo di LaTeX. Si procede come descritto nel paragrafo 7 della documentazione di fancyhdr dove viene esemplificato come modificare lo stile `plain`; qui ci muoviamo in modo simile per modificare lo stile `empty`:

```
\fancypagestyle{empty}{%
\fancyhf{}
\fancyhead[L]{\griglia}
\renewcommand{\headrulewidth}{0pt}
\renewcommand{\footrulewidth}{0pt}}
```

In modo analogo si procede con gli altri stili di pagina.

Nel preambolo o in un file di macro personali si definiscono la variabile booleana e il comando `\griglia`:

```
\newif\ifgriglia \grigliafalse
\newcommand\griglia{%
```

1. I comandi `\makeatletter` e `\makeatother` servono per poter usare il segno @ come se fosse una lettera dell'alfabeto; sono necessari solo se la ridefinizione viene eseguita nel preambolo del documento; *non sono necessari* se inseriti in un file di macro personali da immettere nel preambolo con `\usepackage`.

```
  \ifgriglia
     \csname @griglia\endcsname
  \fi}
```

Il comando `\@griglia` è quello che dovrà disegnare la griglia di sfondo; il test `\ifgriglia` permette di decidere se disegnare o non disegnare la griglia di sfondo; tocca all'operatore impostare `\grigliatrue` dopo l'istruzione `\begin{document}`, almeno all'inizio del documento o, al massimo, subito dopo aver composto il frontespizio.

Il comando `\@griglia` è più delicato, nel senso che la sua definizione va ritardata all'inizio del documento, cioè quando la classe e i vari pacchetti che vengono usati sono già stati tutti caricati:

```
\AtBeginDocument{%
    \normalfont
    \newdimen\scartamento
    \scartamento=\baselineskip
    \newcount\righe
    \righe=\numexpr(\dimexpr(%
       \textheight-\topskip
         +0.5\scartamento)/\scartamento)

    \newcommand{\@griglia}{{%
    \unitlength=\scartamento
    \raisebox{%
       -\dimexpr\headsep+\topskip}%
       [0pt][0pt]{color{red}%
       \picture(0,0)
         \multiput(0,0)(0,-1){\righe}%
           {\rule{\textwidth}{0.4pt}}
       \endpicture}}}
}
```

Caricando il pacchetto color o, meglio, xcolor, la griglia si può disegnare in rosso, in modo da vedere meglio se le righe del testo composto giacciono con precisione sulle righe della griglia.

Con questo piccolissimo aiuto, diventa relativamente facile in fase di revisione del documento finito, aggiungere il comando `\grigliatrue` e aggiustare le dimensioni delle immagini o aggiungere degli spazi verticali positivi o negativi prima o dopo gli oggetti di grandi dimensioni che disturbano la composizione in griglia. Terminato il lavoro di messa a punto, si reimposta `\grigliafalse` e si ricompila per ottenere il documento finito senza che la griglia resti visibile mediante il disegno sullo sfondo.

5 Conclusione

La composizione in griglia è difficile. Dopo questa affermazione quasi lapalissiana, con LaTeX c'è modo di fare le cose per bene, anche se, purtroppo, ancora in modo manuale. Tuttavia l'idea di inserire nello sfondo il disegno della griglia aiuta moltissimo e lavorando a mano si può esercitare il proprio giudizio estetico per valutare come e cosa correggere per realizzare la composizione in griglia.

Le macro suggerite nell'ultimo paragrafo sono semplici e dovrebbero essere alla portata di un utente di LaTeX che non sia un principiante assoluto, ma non richiedono nessuna conoscenza particolare oltre a quanto si trova scritto nel manuale di Leslie Lamport, (Lamport, 1994), o nella sua traduzione commentata contenuta nell'apposita guida tematica (Beccari, 2107) liberamente scaricabile dall'apposita sezione del Forum GuIt.

Riferimenti bibliografici

Bazargan, K. e Radakrishnan, CV (2006). «Removing vertical stretch – mimiking traditional typesetting with TeX». *ArsTEXnica*, (2), pp. 48–53.

Beccari, C. (2107). «Il LaTeX reference manual commentato». Documento PDF. http://www.guitex.org/home/images/doc/GuideGuIT/latexhandbookcommentato.pdf.

Lamport, L. (1994). *LaTeX – User's guide and reference manual*. Addison Wesley Publishing Company, Reading, MA, 2ª edizione.

Mittelbach, F. (2000). «Formatting documents with floats – A new algorithm for LaTeX 2ε». *TUGboat*, **21** (3).

Pignalberi, G. (2005). «Della produzione di una rivista in LaTeX». Documento PDF. http://www.guit.sssup.it/guitmeeting/2005/articoli/pignalberi.pdf.

River Valley Technologies (2009). «grid.sty – Manual and examples». PDF document. Leggibile co `texdoc grid`.

▷ Claudio Beccari
claudio dot beccari at gmail dot com

A Database Experiment With LuajitLaTeX

Roberto Giacomelli

Sommario

Oggigiorno i database sono tra le risorse più importanti in ogni moderna organizzazione. Molti sforzi vengono fatti affinché i dati siano sicuri, affidabili, strutturati tramite relazioni e immediatamente disponibili.

In ogni lavoro di produzione, i documenti contengono ogni sorta di dati che, nonostante tutto, vengono digitati a mano più e più volte, o copiati e incollati, oppure processati da strumenti esterni.

Nel mondo TeX non esistono database con cui lavorare — TeX non è un *reporting tool* — tuttavia le cose sono cambiate grazie all'arrivo della nuova coppia LuaTeX e LuajitTeX.

Questo lavoro è una guida per l'utente a questo nuovo e potenziato mondo TeX spinto dal linguaggio Lua.

Abstract

Nowdays databases are among the most important resources in any modern organization. A lot of effort is done to keep data safe, reliable, structured by means of relationships and instantly available.

Then in every production work, the related documents have plenty of different kind of data, but they are likely to be keyed again and again by hand or copy/pasted or processed by external tools.

In the TeX world there aren't databases to work with—TeX is not a reporting tool—but things have been changed thanks to the new siblings LuaTeX and LuajitTeX.

This paper is a user's guide to that new enhanced TeX world powered by Lua.

1 Motivazione

Al GUIT meeting 2015 di Trento presentai assieme a Gianluca Pignalberi un articolo (Giacomelli e Pignalberi, 2015) che trattava del problema di generare report con il sistema TeX estraendo i dati — per esempio da database — e costruendone un file sorgente pronto da compilare.

Nelle conclusioni di quel lavoro si accennava alla possibile ulteriore esplorazione di soluzioni basate sul nuovo motore di composizione LuaTeX in grado di connettersi *direttamente* a sorgenti dati relazionali attraverso apposite librerie in Lua. Ebbene, con questo lavoro raccolgo quel testimone.

2 Un problema, più soluzioni

Lo scenario che immagino per questa discussione è quello aziendale o professionale nel quale un gran numero di attività coinvolgono la produzione di documentazione contenente *dati*, per la quale l'utente candida il sistema TeX al ruolo di strumento di reporting di elevata qualità in grado di facilitare la creazione di contenuti in un quadro in continua evoluzione.

2.1 Cambiamenti ed evoluzione dati

Nel contesto aziendale o professionale molto spesso i dati vengono memorizzati in *database relazionali*. Questa tecnologia è tra le più affidabili tra quelle oggi disponibili per la gestione dell'informazione digitale. Essa si basa sui vincoli di relazione tra insiemi omogenei detti *tabelle*. Banche, istituti di ricerca, industrie, amministrazioni pubbliche gestiscono ogni giorno grandi quantità di dati in modo affidabile e sicuro evitando gli enormi problemi organizzativi attribuibili allo spezzettamento dei *repository* informativi tra diversi settori dell'organizzazione.

Spesso i database sono centralizzati in server Internet e perciò raggiungibili con i più vari dispositivi e praticamente ovunque. Ciò ha reso addirittura possibile un intero nuovo panorama tecnologico conosciuto sotto il nome di *cloud*.

2.2 TeX e le tecnologie cloud

Il sistema TeX può essere utilmente impiegato per realizzare documenti di elevata qualità tipografica che incorporano dati provenienti da database. Lo si può fare con due modalità automatiche diverse anche se dopo tutto concettualmente simili:

1. estraendo i dati dal database tramite un programma appositamente impostato dall'utente che genera file sorgenti compilabili dal sistema TeX in documenti PDF;
2. compilando un opportuno file sorgente con LuaTeX o LuajitTeX contenente le istruzioni per estrarre i dati dal database e inviarli alla composizione tipografica per la produzione del documento PDF.

Il metodo più avanzato per implementare la soluzione 1 è usare una libreria di *templating* che adopera file di testo con il ruolo di *modelli* di documento. I template contengono parti testuali fisse e campi variabili e l'articolo citato mostra esempi pratici in Lua e Python per *unire* i dati estratti dal database con i template e spiega come sia

possibile strutturare questi modelli tramite l'inclusione di template in altri template o come derivare un template da un altro tramite l'ereditarietà per produrre documenti di complessità arbitraria.

Nella soluzione 2 il template è sostituito dal sorgente TEX. In qualche modo si dovrà astrarre l'informazione dei campi variabili attraverso nomi come nel templating, e si dovrà preparare un insieme di macro utente per l'interrogazione del database e la formattazione dei dati.

Le due implementazioni sono a ben vedere concettualmente simili — e possono essere addirittura mescolate insieme, per esempio un template genera un sorgente che se compilato compone il documento interrogando autonomamente il database — tuttavia nella pratica esse risultano essere assai diverse in particolare per le potenzialità di sviluppo tipografico dei report a opera dell'utente.

L'obiettivo di questo lavoro è esplorare il modo diretto di accesso ai database e valutare le differenze tra le due tecniche di reporting con riferimento ai seguenti aspetti:

- tecnologico,
- qualità e complessità dei report,
- esperienza d'uso dell'utente.

Nell'articolo verranno fornite le istruzioni per configurare il sistema all'esecuzione degli esempi di codice, perché sia possibile toccare con mano le procedure, comprendere meglio il codice e acquisire le basi per implementare i propri progetti.

3 Requisiti di sistema e librerie

Innanzi tutto occorre disporre di una distribuzione TEX recente — diciamo una TeX Live 2016 o successiva — così da poter compilare i sorgenti con i motori di composizione che incorporano Lua, e perciò in grado di eseguire connessioni a database e *query* scritte nel linguaggio SQL.

Per approfondire il linguaggio SQL e la progettazione dei database sono disponibili tantissimi libri. Un testo di argomento generale e completo per gli scopi che ci si prefigge è "Basi di dati" (Atzeni *et al.*, 2014) di carattere universitario. Invece, un elenco di libri di argomento specifico riguardanti il database manager SQLite è riportato nel sito ufficiale del progetto alla pagina `https://www.sqlite.org/books.html`.

Per imparare la programmazione in Lua può essere studiato il libro dell'autore principale del linguaggio stesso, Roberto Ierusalimschy (Ierusalimschy, 2013), tra l'altro esso stesso composto con LATEX.

Nella sezione seguente giustificherò la scelta del database manager SQLite e descriverò in dettaglio come predisporre il proprio sistema desktop installando le librerie condivise e i pacchetti all'interno della distribuzione TeX Live.

4 Lavorare con SQLite3

SQLite è una libreria scritta in C con licenza di pubblico dominio, utilizzata da numerosissime applicazioni, compreso il browser Firefox e le app di Android, che opera su un unico file per ciascun database. SQLite non è quindi disponibile come servizio di rete perché progettato per un uso desktop su singola postazione.

SQLite è la scelta perfetta per applicazioni locali. Non è richiesta alcuna configurazione, l'affidabilità è provata, e sono disponibili *driver* per la connessione praticamente per tutti i linguaggi di programmazione.

La struttura relazionale composta dall'insieme delle tabelle si può eventualmente migrare facilmente verso altri database di rete — per esempio PostgreSQL — una volta verificate le differenze con il dialetto SQL del server di destinazione.

4.1 Necessità e applicazioni

Immaginiamo per esempio le procedure necessarie per l'emissione delle fatture in uno studio professionale. Lo strumento più usato per produrre questo tipo di documenti è senza dubbio il foglio di calcolo, soluzione semplice da implementare ma che non è scalabile, ovvero non riesce a funzionare bene al crescere delle necessità.

Un applicativo basato su database invece si adatta nel tempo alle mutate esigenze basandosi sulla struttura relazionale dei dati del problema in costante evoluzione.

Non è quindi solo una questione quantitativa in relazione alla mole dei dati da gestire, ma soprattutto è importante che le informazioni siano coerenti nei vincoli di relazione e che i dati siano resi disponibili efficacemente.

4.2 SQLite, un database dinamico

La caratteristica principale di SQLite di cui tener conto è questa: in registrazione non esiste il controllo di corrispondenza tra il tipo di dato e quello dichiarato per la tabella, proprio come accade per i linguaggi dinamici[1]. Non si tratta quindi di uno svantaggio o di una limitazione ma di una scelta di progetto coerente con l'ambiente di sviluppo.

Tuttavia è consigliabile, definendo gli attributi, rispettare i tipi dei dati in modo da obbligarci a sviluppare una struttura relazionale coerente all'informazione del mondo reale, garantendoci al tempo stesso una più semplice e sicura possibilità di migrazione su database server a controllo statico del tipo.

1. I linguaggi di scripting come Lua e Python incorporano il concetto di tipo di dato ma solamente in fase di esecuzione, mentre quelli a tipizzazione statica come C o Rust, eseguono il controllo dei tipi in fase di compilazione. Per questi ultimi linguaggi non è possibile assegnare a un oggetto un tipo che sarà noto solo a run time e perciò sono definiti linguaggi statici.

4.3 Installazione di SQLite3

Su sistemi Debian e derivati, compreso Ubuntu, l'installazione corrisponde al comando:

```
$ sudo apt-get install sqlite3
```

sono inoltre necessari i file di libreria a caricamento dinamico — detti *shared libraries* — presenti nel pacchetto di sviluppo, perciò diamo anche il comando:

```
$ sudo apt-get install libsqlite3-dev
```

Per le altre distribuzioni Linux non è difficile scoprire i comandi equivalenti a questi per il package manager relativo.

Su Windows, l'installazione consiste sostanzialmente nella copia di due file da scaricare dalla pagina di download ufficiale del progetto, uno è la libreria a caricamento dinamico in grado di eseguire operazioni sul database, di estensione `dll`, acronimo di dynamic linked library, e l'altro è l'eseguibile `sqlite3`, un command-line tool che interpreta l'SQL in un ambiente interattivo di tipo REPL[2].

Più precisamente, i file si scaricano dai link della sezione *Precompiled Binaries for Windows* della pagina web ufficiale `https://sqlite.org/download.html`. Lì trovate i file compressi degli eseguibili a corredo e la libreria dinamica per i sistemi a 32 o a 64 bit — scaricate quello corrispondente alla vostra versione di Windows.

Dopo aver decompresso i file, sistemateli in una directory — per esempio `C:\sqlite` — e aggiungetela alla variabile di sistema `PATH` in modo che gli eseguibili e la libreria siano *visibili*.

Su sistemi OS X scaricate il file dalla stessa pagina ma dal link della sezione *Precompiled Binaries for Mac OS X (x86)*.

Per controllare che tutto funzioni, è sufficiente lanciare il programma CLI[3] `sqlite3` in una finestra di console. Al prompt proposto dal comando in esecuzione, digitate `.help` per consultare il testo di aiuto ed `.exit` per uscire.

Naturalmente esistono anche dei programmi grafici per interagire con i database SQLite attraverso il mouse. Consiglio l'ottimo SQLite Manager — un add-on per Firefox — ma ce ne sono tanti altri come *DB Browser for SQLite* per OS X.

4.4 I fiumi italiani

Produco qui un esempio semplice ma non banale allo scopo di mostrare al lettore in cosa consiste lo sviluppo di un database personale, ben conscio del fatto che si tratta di un'attività che richiede studio ed esperienza, comunque non impossibili da raggiungere per gli utenti compreso per quelli del sistema TEX.

2. REPL è l'acronimo di read–eval–print loop.
3. CLI è l'acronimo di Command Line Interface.

Consideriamo alcuni dati sui fiumi italiani, iniziando con un database di un'unica tabella contenente i campi `nome` e `lunghezza` (misurata in km).

L'istruzione SQL che la crea è la seguente:

```
CREATE TABLE fiumi (
    nome      VARCHAR(64) NOT NULL,
    lunghezza INTEGER
);
```

Nel codice SQL ho imposto che il campo `nome` non possa essere nullo perché se mancasse il nome del fiume mancherebbe la principale informazione di riconoscimento e i campi restanti non avrebbero senso. In altre parole, ho espresso un vincolo di esistenza imprescindibile per la relazione `fiumi`.

Vedremo poi come inserire ulteriori relazioni più complesse come per esempio per ricavare l'elenco delle regioni italiane attraversate da ciascun fiume.

Tra le molte diverse modalità di esecuzione svolgerò l'esempio pratico dal prompt di `sqlite3` perché preferibile ai fini didattici grazie al modo interattivo di esecuzione delle query.

Mano a mano che diverrete più esperti preferirete forse il modo grafico oppure quello di inserire il codice SQL in un file di testo per farlo poi interpretare dall'utility `sqlite3` stessa, tutto in una volta.

Per creare il nuovo database, all'interno di una finestra di terminale, dopo aver impostato la directory di lavoro in cui verrà salvato il file, si lancia il comando:

```
$ sqlite3 fiumi.db
```

Se il file `fiumi.db` non esiste, esso verrà creato e aperto come database attivo. Se esiste verrà solamente aperto stabilendone la connessione.

All'interno della sessione interattiva a linea di comando, qualsiasi istruzione SQL avrà effetto sul database corrente. Per accertarci che sia effettivamente così, digitiamo il comando puntato `.databases`:

```
sqlite> .databases
seq name  file
--- ----- ----------------------------------
0   main  /home/roberto/testsqlite/fiumi.db
```

Digitiamo ora direttamente il codice SQL per creare la nostra tabella:

```
sqlite> CREATE TABLE fiumi (
   ...> nome VARCHAR(64) NOT NULL,
   ...> lunghezza INTEGER
   ...> );
sqlite>
```

Ricordiamoci che nella sintassi SQL è obbligatorio separare le definizioni dei campi con una virgola e inserire il punto e virgola finale. Come avrete notato l'utility interattiva non torna al prompt finché l'istruzione digitata su più linee non è terminata.

Passiamo ora a popolare la tabella appena creata con i dati dei primi dieci fiumi più importanti. L'istruzione è la seguente, con ciascuna tupla separata da una virgola e il solito punto e virgola finale di chiusura:

```
INSERT INTO fiumi VALUES
('Oglio',  280),
('Adige',  410),
('Tanaro', 276),
('Reno',   210),
('Tevere', 405),
('Po',     652),
('Piave',  220),
('Adda',   313),
('Ticino', 248),
('Arno',   241);
```

Sempre in modalità interattiva, eseguiamo ora la query dei dati appena inseriti ordinando le tuple in senso decrescente secondo la lunghezza dei fiumi, e richiedendo la stampa nel modo a colonne con il comando puntato `.mode` e l'opzione `column`):

```
sqlite> .mode column
sqlite> SELECT * FROM fiumi
   ...> ORDER BY lunghezza DESC;
Po          652
Adige       410
Tevere      405
Adda        313
Oglio       280
Tanaro      276
Ticino      248
Arno        241
Piave       220
Reno        210
```

Come esercizio si provi a inserire un fiume con nome pari a `NULL` per verificare se effettivamente viene sollevato un errore, e a scrivere la query per calcolare la somma delle lunghezze dei fiumi usando la funzione `sum()` sul campo `lunghezza`.

4.5 Quali sono le regioni attraversate?

L'idea è quella di creare una nuova tabella che rappresenti le regioni e una ulteriore tabella che rappresenti le relazioni tra fiumi e regioni. Per rendere le relazioni tra elementi univoche si utilizza generalmente una *chiave primaria*, cioè un campo o un insieme di campi in una tabella, i cui valori non si ripetono e non sono mai nulli.

La tabella `fiumi` si modifica in questo modo:

```
CREATE TABLE fiumi (
    id INTEGER PRIMARY KEY,
    nome VARCHAR(64) NOT NULL,
    lunghezza INTEGER,
    CHECK (lunghezza > 0)
);
```

La nuova tabella `regioni` sarà presumibilmente la seguente:

```
CREATE TABLE regioni (
    id INTEGER PRIMARY KEY,
    nome VARCHAR(64) NOT NULL
);
```

e la tabella di relazione `passa_da` potrebbe essere:

```
CREATE TABLE passa_da (
    id_fiume INTEGER NOT NULL,
    id_regione INTEGER NOT NULL,

    PRIMARY KEY (id_fiume, id_regione),

    FOREIGN KEY (id_fiume)
    REFERENCES fiumi (id),

    FOREIGN KEY (id_regione)
    REFERENCES regioni (id)
);
```

sulla quale è presente il vincolo di chiave primaria a doppio campo che esprime la condizione biunivoca che un fiume non attraversi la stessa regione più di una volta e, viceversa, che una regione non sia attraversara dallo stesso fiume più di una volta.

Completiamo la definizione richiedendo che effettivamente i valori interi contenuti nei record di questa tabella siano effettivamente identificatori presenti nelle tabelle `fiumi` e `regioni`, e questo è espresso nelle ultime 4 righe di codice SQL.

Per ragioni di compatibilità con versioni precedenti, in SQLite occorre attivare esplicitamente il controllo per i vincoli di chiave esterna attraverso l'esecuzione del comando seguente:

```
sqlite> PRAGMA foreign_keys = ON;
```

questa operazione va *sempre* fatta quando si vuole inserire dati nel database.

Inserendo i dati, saremo in grado di costruire la tabella 1 riportata fuori testo nella prossima pagina, eseguendo la query con il doppio join:

```
SELECT
fiumi.id        AS num,
fiumi.nome      AS fiume,
fiumi.lunghezza AS len,
regioni.nome    AS regione
FROM fiumi, regioni, passa_da
WHERE
passa_da.id_fiume = fiumi.id AND
passa_da.id_regione = regioni.id
ORDER BY fiumi.lunghezza DESC;
```

La query si può inserire in modo permanente nel database sotto forma di *vista*. Lo scopo delle viste è astrarre l'applicazione dai dettagli. Le viste tendono infatti a mantenere la stessa struttura verso le applicazioni nonostante le tabelle sottostanti cambino per il mutare dell'informazione reale rappresentata.

Questa query trasformata in vista può essere ulteriormente migliorata scrivendo questa istruzione SQL in cui la funzione scalare predefinita `group_concat()` concatenerà i nomi delle regioni in una lista separata da virgole:

TABELLA 1: Una tabella ottenuta con la compilazione di un sorgente LuajitLaTeX nel quale il codice esegue una query verso un database SQLite3, come spiegato alla sezione 7.1.

N.	Fiume	Lunghezza (km)	Regioni attraversate
1	Po	652	Piemonte, Lombardia, Emilia-Romagna, Veneto
2	Adige	410	Trentino-Alto Adige, Veneto
3	Tevere	405	Emilia-Romagna, Toscana, Umbria, Lazio
4	Adda	313	Lombardia
5	Oglio	280	Lombardia
6	Tanaro	276	Piemonte, Liguria
7	Ticino	248	Svizzera, Piemonte, Lombardia
8	Arno	241	Toscana
9	Piave	220	Veneto
10	Reno	210	Toscana, Emilia-Romagna
11	Sarca-Mincio	203	Trentino-Alto Adige, Veneto, Lombardia
12	Volturno	175	Molise, Campania
13	Brenta	174	Trentino-Alto Adige, Veneto
14	Secchia	172	Emilia-Romagna, Lombardia
15	Ofanto	170	Campania, Basilicata, Puglia
16	Tagliamento	170	Friuli-Venezia Giulia, Veneto
17	Ombrone	161	Toscana
18	Chiese	160	Trentino-Alto Adige, Lombardia
19	Dora Baltea	160	Valle d'Aosta, Piemonte
20	Liri-Garigliano	158	Abruzzo, Lazio, Campania

```
CREATE VIEW vw_passa_da AS
SELECT
fiumi.nome      AS fiume,
fiumi.lunghezza AS len,
group_concat(regioni.nome, ", ") AS regioni
FROM fiumi, regioni, passa_da
WHERE
passa_da.id_fiume = fiumi.id AND
passa_da.id_regione = regioni.id
GROUP BY fiumi.id
ORDER BY fiumi.lunghezza DESC;
```

4.6 LuajitTeX

Entra finalmente in scena il protagonista: LuajitTeX, il nuovo motore di composizione identico a LuaTeX tranne per il fatto che anziché contenere l'interprete tradizionale Lua, contiene LuaJIT.

LuaJIT è un interprete Lua a cui sono state aggiunte alcune funzionalità, in particolare quella per cui il codice viene compilato con tecniche dinamiche dette *just in time*. Queste tecniche di ottimizzazione hanno per effetto il miglioramento dei tempi di esecuzione.

Devo però chiarire il perché, trattandosi di linguaggio interpretato, ho comunque parlato di *compilazione* per Lua, termine riservato ai linguaggi il cui codice prima dell'esecuzione viene tradotto in codice macchina. Ogni volta che Lua esegue un blocco di codice non lo fa direttamente: esegue prima il processo di compilazione che produce una forma virtuale di linguaggio macchina chiamato *bytecode*, poi esegue il bytecode stesso.

Maggiori informazioni sulla tecnologia di compilazione just-in-time possono essere reperite nell'articolo di Luigi Scarso (SCARSO, 2013), dedicato a LuajitTeX.

LuaJIT comprende anche l'estensione `ffi`, *foreign function interface*, che rende molto semplice il binding con librerie esterne rispetto a quanto si può fare con Lua, non richiedendo la scrittura e la compilazione di codice in linguaggio C. Spero di ritornare sull'argomento in un prossimo articolo.

Per generare una connessione a un database SQLite da Lua o da LuaJIT — e quindi anche dai corrispondenti motori di composizione in dotazione al sistema TeX — il modo più efficace è quello di creare un binding, ossia un componente software che rende possibile l'esecuzione di funzionalità contenute nella libreria dinamica di gestione del database scritta in C, all'interno di un componente esterno, nel nostro caso uno script in Lua.

E proprio grazie alle funzionalità `ffi` che in LuaJIT è possibile scrivere il binding semplicemente digitando le *firme* delle funzioni della libreria ospite, anzi, non è nemmeno necessario farlo perché lo ha già fatto Stefano Peluchetti nell'ambito del suo progetto SciLua.org con il modulo che egli ha chiamato `LJSQlite3` e che trovate alla pagina web `http://scilua.org/ljsqlite3.html`.

Anche per Lua esiste il binding grazie al progetto Kepler — indirizzo web `http://keplerproject.github.io/luasql/` — che si concretizza nel modulo LuaSQL capace di connettersi a database Oracle, MySQL, SQLite, Firebird e PostgreSQL e a quelli che implementano i protocolli ODBC e ADO.

LuaSQL dovrebbe essere la scelta da privilegiare perché al progetto Kepler appartengono gli stessi autori di Lua, e non solamente perché è sicuramente più stabile del piccolo modulo `LJSQLite3`.

Tuttavia ho scelto per i miei progetti di gestione dati con SQLite, proprio il pacchetto `LJSQLite3` per una ragione del tutto casuale: quando iniziai nel 2012, sul mio computer non funzionava bene il gestore di pacchetti Luarocks e non riuscii a configurare LuaSQL nemmeno manualmente. Di contro, l'installazione di `LJSQLite3` fu banale: bastò copiare i file in una sottocartella dove si trovava installato LuaJIT.

In disaccordo con la scelta più logica, preferisco qui puntare su moduli software più esotici certo, ma molto veloci nell'esecuzione e d'immediata installazione.

Altro progetto interessante è LuaSQLite3 alla pagina web http://lua.sqlite.org/index.cgi/index che propone una libreria che si basa sul binding diretto tra Lua e SQLite. Ho scoperto questo modulo durante la stesura dell'articolo consultando il sito web che propone una dettagliata pagina dedicata alla documentazione che mostra chiaramente la mappatura delle funzioni dell'`api` di SQLite, ma non l'ho né installato né provato. Esso dovrebbe poter essere utilizzabile anche in LuaTEX e, per come è costruito, dovrebbe offrire una buona velocità di esecuzione assieme alla completezza di funzioni.

4.7 La ricerca dei file

Abbiamo ancora un particolare che riguarda la distribuzione TeX Live e che ci servirà tra poco per capire come rendere raggiungibili i file delle librerie dai programmi di composizione.

Quando un programma della famiglia TEX cerca un file lo fa attraverso l'utility `kpathsea` che a sua volta legge file binari chiamati database `ls-R` che racchiudono tutta la rappresentazione ad albero di una directory nel file system della ditribuzione.

La ricerca del percorso di un file, per esempio un pacchetto LATEX, avviene sfruttando solamente i database `ls-R` delle directory specificate dall'utente attraverso i file di configurazione `texmf.cnf`.

Sapendo che per `YYYY` s'intende l'anno di uscita della distribuzione, il file `texmf.cnf` principale si trova nel percorso:

```
texlive/YYYY/texmf-dist/web2c/texmf.cnf
```

Leggerne i commenti con un po' di pazienza, permette di capire come sono determinate le directory di ricerca: attraverso variabili e l'espansione di espressioni.

Ci interessano particolarmente le linee 448 e 449 — numeri di linea per il file `texmf.cnf` della versione 2016 — che riporto di seguito su tre linee anziché su due, tra i margini di colonna:

```
% Architecture independent executables.
TEXMFSCRIPTS =
    $TEXMF/scripts/{$progname,$engine,}//
```

Si tratta della definizione di dove nel file system possono essere reperiti file Lua da parte dei programmi di TeX Live.

Tabella 2: Directory contenute nella variabile `$TEXMF` per la distribuzione TeX Live dell'autore. I doppi punti esclamativi iniziali del percorso impongono che verranno ricercati i file solamente tra quelli indicizzati nel database `ls-R` e mai sul disco. Nelle prime righe sono riportati i percorsi nell'albero principale, nella riga centrale il percorso dell'albero locale, e nelle ultime righe quelle dell'albero personale.

Percorsi contenuti nella variabile `$TEXMF`
`!!/usr/local/texlive/2016/texmf-config`
`!!/usr/local/texlive/2016/texmf-var`
`!!/usr/local/texlive/2016/texmf-dist`
`!!/usr/local/texlive/texmf-local`
`/home/roberto/.texlive2016/texmf-config`
`/home/roberto/.texlive2016/texmf-var`
`/home/roberto/texmf`

La variabile `$TEXMF` contiene molte directory riportate nella tabella 2, come è possibile verificare dando in una finestra di console il comando:

```
$ kpsewhich --expand-var '$TEXMF'
```

L'albero locale della TeX Live rende disponibili i pacchetti per tutti gli utenti che accedono al computer, perciò è ragionevole installare i file di libreria in questa parte del file system TDS. Per di più quando si aggiorna la distribuzione con quella dell'anno successivo, l'albero locale non viene alterato dal procedimento ed è di nuovo funzionante con la nuova TeX Live.

In definitiva, uno script per LuajitTEX verrebbe ricercato nel percorso (da leggere come se fosse solo su una riga):

```
!!/usr/local/texlive/texmf-local/
    scripts/luajittex
```

I doppi punti esclamativi indicano che la ricerca dei file verrà effettuata solamente tra i percorsi indicizzati nel database `ls-R`, che va quindi rigenerato dopo aver copiato i file nell'albero locale con il comando `mktexlsr`.

Si lascia al lettore il compito di convertire i percorsi in quelli equivalenti per il sistema operativo Windows o per OS X.

4.8 Installare `LJSQLite3` per LuajitTEX

Completando le operazioni descritte in questa sezione si avrà l'accesso a database SQLite direttamente dall'interno di un sorgente TEX. La cosa dovrebbe fare un certo effetto.

Del resto Lua introduce nel mondo tipografico di TEX possibilità davvero innovative che lasciano immaginare sviluppi futuri a dir poco spettacolari.

I passi dell'installazione sono i seguenti:

1. scaricare il pacchetto `LJSQLite3` sotto forma di file compresso dal repository GitHub all'indirizzo https://github.com/stepelu/lua-ljsqlite3, per un peso inferiore a 10KB.

Molto comodo è ricorrere al pulsante "Clone or download" che si trova sulla destra della pagina web;

2. decomprimere e rinominare la cartella di `LJSQLite3` in `ljsqlite3` con sole lettere minuscole; all'interno della cartella rinominare il file `init.lua` in `ljsqlite3.lua`;

3. scaricare il file del modulo `lua-xsys` dalla pagina GitHub `https://github.com/stepelu/lua-xsys`, una dipendenza del pacchetto precedente che ne incorpora una ulteriore chiamata `templet`;

4. rinominare allo stesso modo la cartella decompressa di `xsys` con lo stesso nome del pacchetto; all'interno rinominiamo il file `init.lua` in `xsys.lua`;

5. rinominiamo anche il file `init.lua` in `templet.lua` all'interno della sub-directory `_dep/templet`;

6. con un editor di testi modificare l'istruzione di caricamento del modulo `templet` all'interno del file `xsys.lua` così rinominato al passo 4: intorno alla riga 18 modificare il codice (riportato su due righe anziché una) da:

```
local templet =
   require "xsys._dep.templet"
```

   in

```
local templet = require "templet"
```

7. copiare nella cartella dell'albero locale `/usr/local/texlive/texmf-local` di TeX Live, o nell'equivalente per il vostro sistema operativo e la vostra configurazione, la cartella `scripts/luajittex` (da creare se non esiste); percorso determinato alla sezione 4.7;

8. eseguire il comando `mktexlsr` per rigenerare i database `ls-R`.

Il cambiamento dei nomi dei file e delle cartelle si rende necessario a causa del differente comportamento delle funzioni di caricamento di LuaJIT e LuajitTEX.

Tutte le operazioni manuali sono semplici anche se sappiamo che la maggior parte di esse non sarebbe stata necessaria per lavorare con LuaJIT mentre lo sono con LuajitTEX.

4.9 LuajitTEX in azione

Per verificare che tutto funzioni è sufficiente compilare con LuajitTEX il sorgente seguente dove l'unica cosa che viene eseguito è il caricamento della libreria `LJSQLite3`:

```
% !TeX program = LuajitTeX

\directlua{
local sql = require "ljsqlite3"
}
\bye
```

Un primo esempio di codice consiste nell'effettuare la connessione al database `fiumi.db` prodotto in precedenza e stampare l'elenco dei fiumi memorizzati nella tabella `fiumi`.

La documentazione del modulo di binding `LJSQLite3` all'indirizzo web `http://scilua.org/ljsqlite3.html`, consente abbastanza semplicemente di ricavare il codice:

```
% !TeX program = LuajitTeX

\directlua{
local sql = require "ljsqlite3"
local conn = sql.open("fiumi.db", "ro")

local query = [[
    SELECT nome, lunghezza FROM fiumi
    ORDER BY lunghezza DESC;
]]
local res, nr = conn:exec(query)
conn:close()
for i = 1, nr do
    tex.print(i)
    tex.print(res.nome[i])
    tex.print(tonumber(res.lunghezza[i]))
    tex.print("")
end
}
\bye
```

e il risultato è:

```
1 Po 652
2 Adige 410
3 Tevere 405
4 Adda 313
5 Oglio 280
6 Tanaro 276
7 Ticino 248
8 Arno 241
9 Piave 220
10 Reno 210
```

Dopo aver caricato la libreria `LJSQLite3`, si crea la connessione dati con la funzione `open()` che accetta come primo argomento il nome del file del database, che in questo caso si trova nella stessa cartella del sorgente, e come secondo parametro la stringa che determina la modalità di apertura. La chiave `"ro"` sta per read-only.

Sull'oggetto *connessione* viene chiamato poi il metodo `exec()` passando come argomento il testo che rappresenta la query in linguaggio SQL. Il metodo restituisce due riferimenti: il result set con tutti i dati richiesti dalla query e il numero dei record trovati.

Si chiude poi la connessione chiamando il metodo `close()`[4], mentre poi un ciclo iterativo provvede a stampare i dati nel documento con chiamate alla funzione `tex.print()`.

La funzione `tonumber()` converte il tipo originario intero `cdata` nel tipo numerico nativo di Lua. Se non la inserissimo la funzione `tex.print()` segnalerebbe un errore perché non è in grado da sola di effettuare la conversione.

4.9.1 Struttura del result set

Di ritorno dalla query, i dati vengono memorizzati in una normale tabella Lua che ha però alcune peculiarità: i dati dell'*i*-esima tupla risultato sono indicizzati sia con la chiave corrispondente al nome della colonna derivante dalla query, in questo caso dalle chiavi `nome` e `lunghezza`, che con il numero della colonna a partire da 1. In modo equivalente sia il nome della colonna sia l'indice intero indicizzano le tuple ordinate.

In altre parole il result set è una tabella contenente le tabelle/array delle *colonne*. Valgono le uguaglianze:

```
res.nome[1] = res[1][1] = "Po"
res.lunghezza[1] = res[2][1] = 652
```

e in generale:

```
res.<colName1>[<i-tupla>] = res[1][<i-tupla>]
res.<colName2>[<i-tupla>] = res[2][<i-tupla>]
...
res.<colNameN>[<i-tupla>] = res[N][<i-tupla>]
```

Il metodo `exec()` accetta anche il parametro opzionale di tipo stringa che configura la struttura del result set. In particolare, se esso contiene il carattere `"h"` che sta per header, l'elemento di indice zero conterrà i nomi delle colonne.

4.10 Compilare con LuajitLATEX

Su alcuni sistemi compreso Linux, in TeX Live non è disponibile il programma LuajitLATEX, l'equivalente di LuaLATEX. I passi che ho seguito per rimediare in modo semplice e veloce non prevedendo la creazione di un eseguibile, sono questi:

1. abilitare il formato modificando il file `fmtutil.cnf` principale (oppure locale) della distribuzione;
2. creare il file precompilato;
3. abilitare lo shell editor alla compilazione con il formato.

Per il passo 1 ho dato il seguente comando da digitare su una sola linea (da adattare se il sistema non è un derivato Debian):

4. Questo passaggio non è essenziale perché Lua chiude la connessione in automatico al termine del blocco di codice un momento prima di distruggere l'oggetto in memoria, ma è buona norma chiudere la connessione nel momento in cui non è più necessaria.

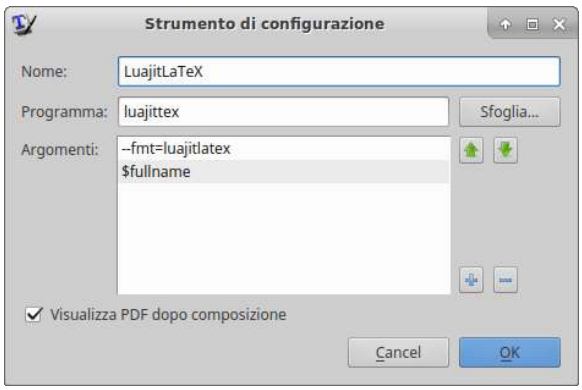


FIGURA 1: Dialogo dello shell editor TeX Works di configurazione del comando per la compilazione di sorgenti nel formato LuajitLATEX sui sitemi che hanno questa configurazione disabilitata. Il comando è poi selezionabile specificandone il nome all'interno delle righe magiche, come mostrato nei sorgenti completi riportati nel testo.

```
$ sudo gedit /usr/local/texlive/
    2016/texmf-dist/web2c/fmtutil.cnf
```

Basta decommentare la riga seguente (riportata qui su due linee) togliendo i caratteri iniziali `#!` che stanno a indicare una configurazione disabilitata:

```
luajitlatex luajittex language.dat,
    language.dat.lua lualatex.ini
```

Per il passo 2 il comando è:

```
$ sudo fmtutil-sys --byfmt luajitlatex
```

Adesso dovrebbe essere possibile compilare sorgenti nel formato LuajitLATEX con il comando:

```
$ luajittex --fmt=luajitlatex <nome-file>
```

Per comodità, questo comando può essere lanciato automaticamente dall'interno di uno shell editor premendo il bottone o la combinazione di tasti prevista. Con TeX Works si configura nel dialogo delle Preferenze, scheda Composizione, un nuovo comando come mostrato in figura 1, il cui nome può essere inserito nelle righe magiche del sorgente. Le istruzioni di lancio del comando comprende l'opzione `-fmt=luajitlatex`.

La verifica può essere fatta sul seguente sorgente:

```
% !TeX program = LuajitLaTeX

\documentclass{article}
\usepackage{booktabs}
\begin{document}

\begin{tabular}{lc}
\toprule
Nome & Lunghezza (km)\\
\midrule
\directlua{
local sql = require "ljsqlite3"
local conn = sql.open("fiumi.db", "ro")

local query = [[
```

```
    SELECT nome, lunghezza FROM fiumi
    ORDER BY lunghezza DESC;
]]
local res, nr = conn:exec(query)
conn:close()
for i = 1, nr do
    tex.print(res.nome[i])
    tex.print("&")
    tex.print(tonumber(res.lunghezza[i]))
    tex.print("\string\\\string\\")
end
}
\bottomrule
\end{tabular}
\end{document}
```

5 La regola aurea

Termino questa sezione introduttiva enunciandovi una regola che osservo senza alcuna eccezione. Non farlo genera inconvenienti.

La regola aurea che l'utente TEX deve rispettare quando si connette a un database è questa:

Connessioni in sola lettura!

In altre parole la regola equivale a usare i motori di composizione LuaTEX o LuajitTEX esclusivamente come programmi di reporting. Le operazioni di popolamento, cancellazione e modifica dovranno essere implementate con altri strumenti.

I dati sono quindi la risorsa principale al centro del sistema, in cui componenti software inviano informazioni verso il database, mentre altri del tutto indipendenti dai precedenti leggono i dati con cui realizzare report.

6 Dati e comandi

Dedicherò questa parte dell'articolo alla ricerca di soluzioni per creare un ponte tra i dati disponibili in Lua e la loro rappresentazione come oggetti tipografici composti, attraverso la definizione di comandi TEX.

Una volta eseguite le query verso il database, i dati risiederanno in memoria come strutture disponibili all'interno di Lua. Essi potranno fluire verso la rappresentazione a stampa nel documento finito per elaborazione TEX, in due modi:

- inserimento di caratteri nello stream di *token*,
- inserimento di oggetti tipografici completi nello stream delle liste principali del *processo visuale* di TEX.

Il primo modo si concretizza nelle varie funzioni `print` del modulo interno `tex` di LuaTEX o LuajitTEX, per esempio come nel seguente sorgente:

```
% !TeX program = LuaTeX

\directlua{
tex.print(math.pi)
}
\bye
```

L'espansione della primitiva `\directlua` è vuota ma l'esecuzione del codice Lua ha in questo caso, l'effetto di inserire le cifre di π — fino a una certa precisione — come se fossero già state presenti nel file sorgente. In altre parole, quella funzione immette caratteri alimentando lo stadio iniziale dell'elaborazione del motore tipografico.

La seconda modalità è molto più sofisticata. Consiste nel creare oggetti Lua chiamati *nodi* equivalenti a quelli presenti nelle liste di composizione dell'ultima fase — detta visuale — quando gli elementi tipografici ormai completi vengono composti sulla pagina.

I nodi devono formare liste concatenate. Ve ne sono molti diversi tipi come glifi, grandezze elastiche, e scatole orizzontali e verticali, che formano quindi catene di oggetti elementari ricchi di dettagli tipografici e costruiti con la precisione del punto scalato, l'unità di misura interna di TEX che corrisponde a una frazione davvero minuscola del millimetro.

Nel seguito utilizzerò il primo modo descritto con la stampa di token, meno sofisticato ma più semplice che non la creazione di oggetti tipografici in Lua, al fine di concentrare l'attenzione sull'elaborazione dei dati e sulla scrittura di macro per l'utente finale. Il lettore può riferirsi al manuale di LuaTEX (The LuaTEX development team, 2016) o ai miei due recenti articoli apparsi nel 2016 su Ars, (Giacomelli, 2016b), (Giacomelli, 2016a) per approfondire la tecnologia dei nodi.

6.1 Espressioni

L'unico oggetto strutturato di Lua — non considerando possibili estensioni scritte in C o anche in altri linguaggi — è la *tabella*, allo stesso tempo un dizionario chiave/valore e/o un array a una dimensione.

La particolare sintassi Lua detta *dot notation* restituisce il valore associato alla chiave stringa con qualifica di identificatore:

```
math.pi == math["pi"]
```

ed è un'espressione. Potremo quindi scrivere la seguente macro `\print` che accetta un testo che corrisponde a una espressione Lua:

```
% !TeX program = LuaTeX

\def\print#1{%
\directlua{tex.print(#1)}}

\print{math.pi}
\bye
```

Con questa semplicissima macro è possibile inserire nel documento qualsiasi espressione Lua e

in particolare, non solo valori associati a chiavi di tabelle, ma anche chiamate di funzione:

```
% !TeX program = LuaTeX

\def\print#1{%
\directlua{
tex.print(#1)
}}

\begingroup
  \catcode`\%=12\relax
  \gdef\prc{%}
\endgroup

% definizione di una funzione
\directlua{
cerchio = {}
cerchio.area = function(r, prec)
    prec = prec or 2
    local fmt = "\prc0."..prec.."f"
    return string.format(fmt, math.pi*r^2)
end
}

\print{cerchio.area(5)}
\print{cerchio.area(15, 6)}
\print{cerchio.area(2) - cerchio.area(1)}
\bye
```

6.2 Stampa condizionale

Per stampa condizionale intendo una macro TEX che stampa il testo che corrisponde al valore di una condizione vera o falsa, la cui definizione di base potrebbe essere la seguente:

```
% !TeX program = LuaTeX

\def\ifprint#1#2#3{%
\directlua{
if #1 then
    tex.print([==[#2]==])
else
    tex.print([==[#3]==])
end
}}

\directlua{x = 5}

\ifprint{x > 10}{%
$x$ maggiore di 10}{%
$x$ non maggiore di 10}
\bye
```

Il primo argomento della macro `\ifprint` è l'espressione Lua che è valutata nel valore booleano vero o falso, il secondo argomento è il testo da stampare se la condizione è vera, e il terzo argomento è il testo da stampare se la condizione è falsa.

Non è necessario racchiudere gli argomenti testuali in apici o doppi apici poiché nel normale meccanismo di sostituzione degli argomenti, TEX ne espanderà il valore all'interno di delimitatori estesi del tipo stringa di Lua — la coppia simmetrica `[==[` e `]==]`.

Interessante notare che l'espansione degli argomenti fa si che questo comando funzioni senza alcun problema:

```
\ifprint{x < 10}{%
$x = \print{x}$ maggiore di 10!}{%
$x = \print{x}$ non maggiore di 10}
```

Infatti la macro condizionale `\ifprint` riceverà gli argomenti già espansi, perciò con il valore della variabile `x` al posto dei token `\print{x}`.

6.3 Stampa se l'espressione è vera

Potremo trovarci a scrivere macro condizionali `\ifprint` per stampare un testo solamente nel caso in cui un oggetto esiste oppure no. Uno dei due argomenti testuali dovrebbe essere un argomento tra graffe vuoto. Tuttavia un comando specifico avrebbe una semantica più significativa.

La definizione seguente è altrettanto semplice di quelle precedenti:

```
% !TeX program = LuaTeX

\def\iftrueprint#1#2{%
\directlua{
if #1 then
    tex.print([==[#2]==])
end
}}

\directlua{x = 5}
\iftrueprint{x > 10}{%
$x$ maggiore di 10}
\bye
```

6.4 Stampa con iteratori

Un iteratore è un modo semplice di elaborare uno alla volta gli elementi di una collezione di dati. In Lua si utilizza il costrutto chiamato *generic for* specificando i nomi delle variabili, che saranno disponibili all'interno del ciclo, e la funzione che restituisce l'iteratore. Traslando il generic for in una macro potremo scrivere:

```
\def\iterprint#1in#2do#3{%
\directlua{
for #1 in #2 do
    tex.print(#3)
end
}}
```

Sfruttando la tecnica ad argomenti delimitati, la macro `\iterprint` accetta come argomento 3 un'espressione che potrà elaborare le variabili di ciclo denominate all'argomento 1, e il cui risultato sarà valutato a ogni iterazione del ciclo definito dall'iteratore dato come argomento 2, per poi essere immesso nel testo del documento.

Se volessimo stampare tutte le coppie chiavi/valore contenute in una tabella potremo usare l'iteratore predefinito `pairs()` che restituisce a

ogni ciclo in prima posizione la chiave e in seconda il corrispondente valore. Il seguente codice stamperà tutti gli oggetti contenuti nella tabella `math` che è il modulo matematico di Lua:

```
% !TeX program = LuaTeX

\def\iterprint#1in#2do#3{%
\directlua{
for #1 in #2 do
    tex.print(#3)
end
}}

\iterprint k, v in pairs(math) do {
    k.." = "..type(v).."\string\\par"
}
\bye
```

Invece di un'espressione è possibile stampare il risultato di una funzione. In questo modo gli elementi della collezione vengono mappati e/o filtrati in una seconda collezione di oggetti. Questo esempio spiega meglio l'operazione:

```
% !TeX program = LuaTeX

\def\itermapprint#1in#2do#3{%
\directlua{
local function tmp (#1)
    #3
end
for #1 in #2 do
    local s = tmp(#1)
    if s then
        tex.print(s)
    end
end
}}

\itermapprint k, v in pairs(math) do {
    if type(v) == "number" then
        return k.." = "..v.."\string\\par"
    end
}
\bye
```

La `\itermapprint` rispetto alla macro precedente esegue il proprio argomento 3 come una funzione con gli argomenti di ciclo. Se l'esecuzione restituisce un valore esso verrà stampato altrimenti si salta alla successiva iterazione.

Questo è proprio quello che succede nel codice di esempio dove solamente gli elementi di tipo numerico della tabella `math` vengono mappati nella stringa field name = numeric value.

La mappatura o il filtraggio possono essere anche espressi come funzioni che accettano un iteratore e ne restituiscono un secondo. Tuttavia ne risulterebbe una sintassi poco chiara perché, per come sono costruiti gli iteratori in Lua, non è possibile concatenarli con la sintassi in dot notation, ma solamente come funzione dentro funzione.

6.5 Includere query

Queste macro possono essere adattate per eseguire query e lasciare che l'utente gestisca con un'espressione i risultati. I comandi ad argomenti delimitati rendono semplice la definizione, e una libreria in Lua può nascondere i dettagli del codice per le interrogazioni pur mantenendo la generalità.

Scriveremo queste funzionalità nella prossima sezione, sviluppando l'esempio della creazione di report rispetto al semplice database sui fiumi.

7 Fiumi

7.1 Da tabella a tabular

Nella sezione 4.4 ho definito un semplice database che contiene dati sui principali fiumi italiani. In particolare esso contiene le lunghezze in km e le regioni attraversate per ciascun fiume. In questa sezione spiegherò come, con le conoscenze acquisite nelle sezioni precedenti, sia possibile ottenere la tabella 1 riportata a pagina 5.

Prepariamo allo scopo alcune funzioni Lua che ci serviranno per connetterci al database e iterare le tuple risultato della query per creare le righe della tabella. Requisito di progetto di queste funzioni è la generalità, cioè dovremo fare in modo che valgano per qualsiasi query su un qualsiasi database SQLite.

Salviamo nella stessa directory del database un file di testo che chiameremo `dblib.lua`, con il codice diviso in quattro parti:

1. caricamento della libreria LuaJIT in una variabile locale;
2. creazione di una tabella di modulo;
3. definizione delle funzioni e dei parametri come oggetti da assegnare a chiavi nella tabella di modulo;
4. istruzione di ritorno della tabella di modulo.

Quando questo file verrà caricato con l'istruzione `require()` all'interno del sorgente LuajitLaTeX, potremmo disporre delle funzionalità definite nella tabella di modulo mentre la libreria LJSQLite3 sarà un oggetto nascosto all'interno della closure d'ambiente.

Per provare che funzioni questa semplice ma efficace soluzione, scriviamo dapprima solamente le funzioni per l'apertura in sola lettura e la chiusura di una connessione a un database, per la nostra libreria:

```
local sql = require "ljsqlite3" -- parte 1
local db = {} -- parte 2

function db:connect(dbpath) -- parte 3
    self.conn = sql.open(dbpath, "ro")
end
```

```
function db:close()
    self.conn:close()
end

return db -- parte 4
```

La notazione detta *method call* è stata introdotta in Lua assieme ai metametodi e alle metatabelle per la programmazione a oggetti. Questa particolare sintassi in realtà è molto semplice perché aggiunge un primo argomento con il riferimento alla tabella stessa con il nome di `self` alla funzione. Si tratta solamente di *zucchero sintattico*, cioè una funzionalità dell'interprete, e non del linguaggio, che semplifica il codice. Le due espressioni seguenti sono del tutto equivalenti:

```
-- method call
db:connect("dbname")

-- equivale a chiamare:
db.connect(db, "dbpath")
```

Con il method call, la funzione `connect()` memorizza all'interno della tabella di modulo stessa il riferimento alla connessione del database, con uno stile non proprio pulito. Dovremo infatti utilizzare pienamente il paradigma a oggetti ma qui ci interessa un codice più semplice possibile per concentrarci sull'obiettivo. Il sorgente LuajitTEX che mostra come usare la libreria appena scritta è il seguente, ipotizzando che il database sia un file chiamato `fiumi.sqlite`:

```
% !TeX program = LuaJitTeX

\directlua{
local dblib = require "dblib"
dblib:connect("fiumi.sqlite")
dblib:close()
}
\bye
```

ed essendo un documento vuoto, nessun PDF sarà composto. Lo scopo è quello di controllare se tutto sia in ordine e funzionante. Passiamo ora a scrivere la funzione centrale: essa dovrà eseguire la query e, compito più complesso, restituire un iteratore sull'insieme delle tuple della tabella risultato.

L'iteratore è un buon modo per rendere semplice l'elaborazione delle tuple. Tecnicamente però, i dati si trovano già tutti caricati in memoria al momento dell'iterazione e perciò non si tratta di un vero *cursore* che invece è una funzionalità del database.

L'implementazione dell'iteratore che useremo tra poco è questa:

```
function next(inv_state, ctrl_var)
    local res = inv_state.rs
    local nr = inv_state.nr
    ctrl_var = ctrl_var + 1
    if ctrl_var <= nr then
        local cols = #res
        local t = {}
        for i = 1, cols do
            local data = res[i][ctrl_var]
            if type(data)  == "cdata" then
                data = tonumber(data)
            end
            t[#t+1] = data
        end
        return ctrl_var, t
    end
end

function db:iter_row(query, nrecords)
    local conn = self.conn
    -- prepared statement:
    local stmt = conn:prepare(query)
    local rs, nr =
        stmt:resultset(nil, nrecords)
    stmt:close()
    return next, {nr = nr, rs = rs}, 0
end
```

La funzione `iter_row()` ha due argomenti, il primo è la stringa contenente il codice SQL che rappresenta la query, il secondo è il numero massimo dei record da restituire in base all'ordinamento stabilito dalla query. Se questo parametro non è fornito vengono restituiti tutti i record.

Dovremo quindi scrivere la query come codice SQL, ma era implicito nelle specifiche di progetto, quando si richiedeva la generalità sull'interrogazione. Sarebbe possibile impiegare un modulo software chiamato ORM acronimo di Object Relational Mapping, per esempio http://fperrad.github.io/lua-CoatPersistent/. Un ORM permetterebbe anche di astrarre il codice anche rispetto al particolare database management system.

Nel dettaglio, all'inizio dell'iterazione la funzione `iter_row()` viene eseguita e i tre valori di ritorno assegnati ai parametri interni del ciclo `for`. Il primo è la funzione da chiamare per ottenere gli elementi, il secondo è lo stato invariante e il terzo un indice di controllo.

Si tratta quindi di uno *stateless iterator* perché la funzione `next()` non fa uso di variabili interne alla closure corrispondente, ma solamente dei tre parametri interni. Se vogliamo, anche il costrutto `for` può essere considerato zucchero sintattico nel senso che esso è equivalente a un opportuno ciclo `while`.

Osserviamo infine che la funzione `next()` prende come argomenti lo stato invariante e l'indice di controllo e restituisce il valore da assegnare all'indice di controllo per il ciclo successivo e le variabili d'iterazione, che in questo caso è la sola tabella array del record corrente.

L'iteratore così definito, rispettando le specifiche previste dal generic for di Lua, può essere impiegato in una delle due macro introdotte nella sezione 6.4. Il sorgente è il seguente:

```
% !TeX program = LuaJitLaTeX
```

```
\directlua{
dblib = require "dblib"
dblib:connect("fiumi.sqlite")
}

\def\iterprint#1in#2do#3{%
\directlua{
for #1 in #2 do
    tex.print(#3)
end
}}

\def\endrow{\string\\\string\\}

\documentclass[margin=2pt]{standalone}
\usepackage{luatex85}
\usepackage{fontspec}
\usepackage{booktabs}
\begin{document}
\small
\begin{tabular}{llcl}
\toprule
N. & Fiume & Lunghezza (km) & Regioni
attraversate\\
\midrule
\iterprint i, row in
    dblib:iter_row(
      [[SELECT * FROM vw_passa_da;]],
      15
    ) do {
    i.."&"..table.concat(row, "&")..
    "\endrow"
    }
\bottomrule
\end{tabular}
\directlua{dblib:close()}
\end{document}
```

Interessante notare che in questo esempio il codice Lua occupa solamente un nome globale, nome che è possibile cambiare e che è stato definito come `dblib`. Ciò praticamente annulla i problemi di collisioni di nomi di variabili globali in documenti complessi in cui si caricano molti pacchetti Lua diversi.

Il pacchetto luatex85 definisce alcune macro per ripristinare i vecchi nomi di primitive che furono cambiati nella versione 0.85 di LuaTEX per rendere più chiaro la distinzione tra motore di composizione e formato di uscita. Lo faccio notare perché una volta che gli Autori dei pacchetti avranno aggiornato il proprio codice, questo pacchetto non sarà più necessario.

Per esempio, se nel nostro sorgente non caricassimo il pacchetto luatex85 riceveremmo un errore di Undefined control sequence per `\pdfpagewidth` utilizzata proprio dalla classe standalone. L'alternativa al pacchetto è copiare le definizioni riportate nel manuale di LuaTEX unicamente per le primitive interessate dalla compilazione del documento.

7.2 TEX si connette a un database

Invece di creare e chiudere la connessione al database attraverso codice Lua diretto come abbiamo fatto nel listato precedente, scriviamo due macro generiche. Stiamo quindi creando diversi strati di funzionalità: una sorta di pacchetto utente per fornire le macro di alto livello e un modulo Lua sottostante scritto in un file dedicato.

Le macro potrebbero essere le seguenti, anche se non contengono alcun controllo sugli errori che qui appensentirebbe la comprensione del codice ma che in realtà è irrinunciabile:

```
\def\dbconnect#1{
\directlua{
local dblib = require "dblib"
dblib:connect([==[#1]==])
}}

\def\dbclose{
\directlua{
local dblib = require "dblib"
dblib:close()
}}
```

La funzione `require()` primitiva di Lua, è utilizzata ogni volta per ottenere il riferimento alla tabella che contiene le nostre funzioni del livello base. Se la libreria è già stata caricata essa risulta memorizzata come riferimento nella tabella `package.loaded` e la funzione `require()` restituirà semplicemente quel riferimento. Ad essere unico non dovrà più essere il nome della variabile globale che contiene il modulo ma solamente il nome del modulo stesso.

7.3 Un miglioramento: funzioni formato

All'interno della macro `\iterprint` nel listato del sorgente precedente che produce la tabella sui primi 15 fiumi italiani, dobbiamo ricreare la riga di testo con tutti i campi separati dal carattere `&` e con il doppio backslash finale. Ciò non è proprio elegante.

In dettaglio, la concatenazione di questi campi avviene nell'argomento 3 della macro `\iterprint`, dove dobbiamo inserire un'espressione Lua. Nulla vieta di sostituire l'espressione con la chiamata a una funzione che fa il lavoro di concatenazione per noi.

La nuova versione del sorgente è questa:

```
% !TeX program = LuaJitLaTeX

\def\dbconnect#1{...} % as before
\def\dbclose{...}

\def\iterprint#1in#2do#3{%
\directlua{
for #1 in #2 do
    tex.print(#3)
end
}}
```

```
\dbconnect{fiumi.sqlite}
\directlua{dblib = require "dblib"}

\documentclass[margin=2pt]{standalone}
\usepackage{luatex85}
\usepackage{fontspec}
\usepackage{booktabs}
\begin{document}
\small
\begin{tabular}{llcl}
\toprule
N. & Fiume & Lunghezza (km) & Regioni
attraversate\\
\midrule
\iterprint i, row in
  dblib:iter_row(
     [[SELECT * FROM vw_passa_da;]],
     15
  ) do {
     dblib.concat("&", dblib.endrow, i, row)
  }
\bottomrule
\end{tabular}
\dbclose
\end{document}
```

La funzione `dblib.concat()` si trova quindi nel file di libreria, dobbiamo solo aggiungerla alla sezione 3 del sorgente `dblib.lua`. La funzione è del tutto generale, essa concatena qualsiasi argomento, una volta specificata la stringa di separazione tra i valori e la stringa finale. Oltre ai primi due argomenti obbligatori riceve un qualsiasi numero di variabili, siano esse valori semplici o tabelle, gestendole ricorsivamente.

Si tratta quindi di una funzione piuttosto sofisticata perchè utilizza due peculiarità di Lua: la *tail call*[5] che offre la sicurezza di chiamate ricorsive che *mai* saturano lo stack e il *three dots* che rappresenta un numero variabile di argomenti.

Grazie all'uso di queste e altre caratteristiche avanzate di Lua, l'utente può disporre di funzioni molto flessibili che in questo caso corrispondono a *funzioni formato*, nel contesto dell'elaborazione dei dati per la loro preparazione alla stampa.

Ecco le aggiunte alla libreria:

```
local function conc_rec(buf, sep, t, index)
    if index > #t then return buf end
    local elem = t[index]
    if type(elem) == "table" then
        buf = conc_rec(buf, sep, elem, 1)
    else
        buf[#buf+1] = elem
        buf[#buf+1] = sep
    end
    -- tail call
    return conc_rec(buf, sep, t, index+1)
end

-- wrapper function:
function db.concat(sep, endstr, ...)
    local vararg = {...}
    local buf = conc_rec({}, sep, vararg, 1)
    buf[#buf] = endstr
    return table.concat(buf)
end

db.endrow = [[\\]]
```

Da notare che se ci fossero dei valori `nil` tra gli argomenti variabili, ciò arresterebbe la scansione al primo di essi e i successivi verrebbero ignorati. Prima di apportare modifiche alle funzioni è necessario decidere se trattare il `nil` con il significato di cella vuota oppure no, ignorandolo e saltando all'elemento successivo.

Il controllo più fine sulla lista degli argomenti variabili può esser fatto per mezzo della funzione Lua `select()`.

7.4 Macro e query

Invece di usare la macro `\iterprint` potremo semplificare il codice accettando come argomento non l'iteratore ma direttamente il testo della query in SQL. Chiamiamo quindi questa nuova macro utente come `\selectprint` a indicare con la parte `select` che dovrà essere fornita una query e con la parte `print` che dovrà essere fornita un'espressione che verrà poi valutata e stampata come token nel documento:

```
\def\selectprint#1in#2;#3{%
\directlua{
local db = require "dblib"
local concat = db.concat
local endrow = db.endrow

for #1 in db:iter_row([[#2]]) do
    tex.print(#3)
end
}}
```

che può essere utilizzata all'interno di un ambiente tabular come:

```
\selectprint i, row in
  SELECT * FROM vw_passa_da LIMIT 15;
  {concat("&", endrow, i, row)}
```

7.5 Da tabella a grafico

Sviluppiamo ora un istogramma per rappresentare le lunghezze dei fiumi memorizzati nel database. Questo nuovo lavoro è una concreta applicazione del concetto di modello/vista, sott'inteso a quello di report definito come un documento che contiene dati provenienti da una fonte esterna.

Useremo il pacchetto pgfplots per comporre il grafico costruendo un sorgente statico per la messa a punto del codice che servirà da modello per la versione che caricherà i dati dal database. Il sorgente modello è il seguente dal risultato riportato in figura 2:

5. Maggiori informazioni su questo tipo di chiamata sono reperibili nel manuale di Lua (Ierusalimschy, 2013), e nei testi che trattano gli argomenti della programmazione funzionale.

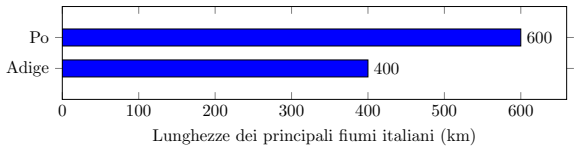


FIGURA 2: Istogramma generato con il pacchetto pgfplots con il codice riportato nel testo per essere utilizzato come base per costruire l'istogramma analogo ma con i dati ricavati da un database.

```
% !TeX program = pdflatex
\documentclass{standalone}
\usepackage{pgfplots}
\pgfplotsset{compat=1.14}

\begin{document}
\begin{tikzpicture}
\begin{axis}[
xbar,
xmin=0,
width=12cm, height=3.5cm,
enlarge y limits=1.0,
xlabel={Lunghezze dei principali fiumi
    italiani (km)},
symbolic y coords={Adige,Po},
ytick=data,
nodes near coords,
nodes near coords align={horizontal}
]
\addplot[draw=black,fill=blue]
coordinates {
(600,Po)
(400,Adige)
};
\end{axis}
\end{tikzpicture}
\end{document}
```

I dati del grafico che devono essere inseriti nel sorgente sono suddivisi in due gruppi: la lista separata da virgole dei nomi dei fiumi da dare all'opzione `symbolic y coord` ordinata dal fiume meno lungo al più lungo, e le coppie di coordinate all'interno del comando `\addplot`, formate dalle coppie ordinate lunghezza e nome fiume. Ricaveremo queste informazioni dal database attraverso query.

Per prima cosa ho messo a punto la query che genera la lista dei nomi dei fiumi a parte, lavorando con il plug-in per Firefox SQLite Manager[6] usando la funzione `group_concat()` su una sub-query[7] che elenca i nomi dei primi 24 fiumi più lunghi ma in ordine inverso, e che richiede a sua volta una query per determinare il numero dei fiumi rimanenti, quelli dalla posizione 24 in poi, da passare al parametro `OFFSET`:

```
SELECT group_concat(nome, ",") FROM (
    SELECT nome FROM fiumi
    ORDER BY lunghezza ASC
    LIMIT 24 OFFSET (
        SELECT count(id) FROM fiumi) - 24
);
```

Poi sono passato alla query che produce la lista delle coordinate, anche questa in sub-query per sfruttare la funzione `group_concat()`:

```
SELECT group_concat(coord , " ") FROM (
SELECT '(' || lunghezza || ',' || nome || ')'
AS coord FROM fiumi
ORDER BY lunghezza DESC
LIMIT 24
);
```

Nel sorgente LuajitLaTeX ho poi inserito il codice SQL all'interno delle macro `\selectprint` a sua volta inseriti in macro utente definite nel preambolo perché il codice sia più ordinato. Il listato completo che genera la figura 3 è risultato così il seguente:

```
% !TeX program = LuaJitLaTeX

% define macro as before

\documentclass[margin=2pt]{standalone}
\usepackage{luatex85}
\usepackage{pgfplots}
\pgfplotsset{compat=1.14}

\newcommand\symbcoord{
\selectprint i, list in
SELECT group_concat(nome, ",") FROM (
    SELECT nome FROM fiumi
    ORDER BY lunghezza ASC
    LIMIT 17 OFFSET (
        SELECT count(id) FROM fiumi
    ) - 17
);{list}}

\newcommand\coords{
\selectprint i, list in
SELECT group_concat(coord, " ") FROM (
SELECT '(' || lunghezza || ',' || nome || ')'
AS coord FROM fiumi
ORDER BY lunghezza DESC
LIMIT 17
);{list}}

\begin{document}
\begin{tikzpicture}
\begin{axis}[
xbar,
bar width=9pt,
width=16cm,
ymin=Ombrone,
ymax=Po,
enlarge y limits=0.035,
xmajorgrids,
```

6. Per editare il codice SQL di una query è molto più comoda la finestra di testo di SQLite Manager che il programma `sqlite3` a riga di comando con cui si lavora solamente una riga alla volta.

7. La funzione `group_concat()` di SQLite non opera quando nella query inseriamo una condizione `LIMIT`, forse perché l'ordine di concatenazione è arbitrario, come si legge nella documentazione di riferimento ufficiale.

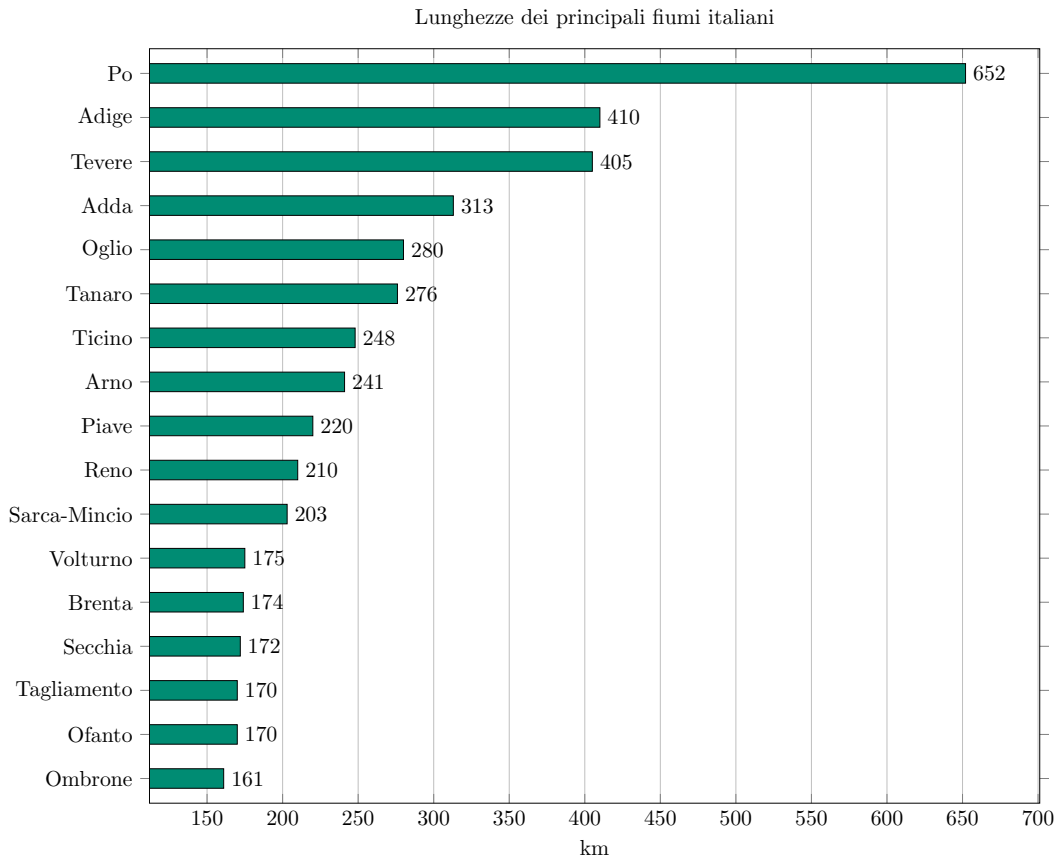


FIGURA 3: Istogramma generato con il pacchetto pgfplots con il codice riportato nel testo che ottiene i dati memorizzati in un database SQLite.

```
title={Lunghezze dei principali fiumi italiani},
xlabel={km},
symbolic y coords/.expanded={\symbcoord},
ytick=data,
nodes near coords
]
\addplot[draw=black, fill=blue!45!green]
coordinates {\coords};
\end{axis}
\end{tikzpicture}
\end{document}
```

Notate l'accortezza di marcare l'opzione `symbolic y coords` con `.expanded` per gestire l'espansione della macro che effettua la query e stampa i token della lista corretta.

Se provaste a modificare il numero n dei fiumi da vuole rappresentare nel grafico, potreste incorrere in un errore di compilazione!

Per esempio, poichè l'ordinamento è richiesto solo per il campo `lunghezza`, con i nostri dati e $n = 15$, l'ultimo fiume del gruppo potrebbe essere per la prima query, che ricava i simboli, il Tagliamento mentre per l'altra query, che restituisce le coordinate, l'Ofanto, perché entrambi di lunghezza di 170 km.

Nella lista delle coordinate avremo quindi un simbolo non dichiarato costringendo pgfplots a interrompere la compilazione.

Questo ci insegna che le query sono strumenti molto potenti ma sono, per così dire, *molto sensibili*.

Per risolvere dovremo includere anche l'ordinamento secondario per nomi. Le nuove query che risolvono il bug si differenziano solo per quest'aggiunta, la prima con ordinamento ascendente e la seconda con quello discendente perché devono produrre ordinamenti opposti, affinché in pgfplots le barre descrescano dall'alto verso il basso:

```
\newcommand\symbcoord{%
\selectprint i, list in
SELECT group_concat(nome, ",") FROM (
    SELECT nome FROM fiumi
    ORDER BY lunghezza ASC, nome ASC
    LIMIT 15 OFFSET (
    SELECT count(id) FROM fiumi) - 15);
{list}}

\def\coords{
\selectprint i, list in
SELECT group_concat(coord , " ") FROM (
SELECT '('|| lunghezza || ',' || nome ||')'
AS coord FROM fiumi
ORDER BY lunghezza DESC, nome DESC
LIMIT 15
);{list}}
```

Questo sorgente inoltre, non è del tutto generale perché da un lato richiede giustamente soltanto

di definire il numero dei fiumi da includere nel grafico, ma dall'altro costringe a passare il primo e l'ultimo fiume dell'intervallo ai parametri `ymin` e `ymax`, parametri utili a evitare che pgfplots inserisca sulla tela del grafico uno spazio bianco troppo grande sopra e sotto le barre dell'istogramma.

7.6 Miglioramenti

Non abbiamo ancora una macro adatta perché `\selectprint`, progettata per ricavare la tabella dei fiumi, presuppone una collezione di dati da stampare. Introduciamo quindi la macro `\selectrowprint` per esprimere il caso frequente di una query che restituisce una sola tupla, senza necessità alcuna d'iterazione.

La macro potrebbe essere la seguente, assegnando direttamente a variabili i campi della tupla risultato:

```
\def\selectrowprint#1<=#2;#3{%
\directlua{
local dblib = require "dblib"
#1 = dblib:rowexec([[#2]])
tex.print(#3)
}}
```

scompare il delimitatore `in` legato al ciclo generic for di Lua, sostituito da una freccia `<=` che può essere letta come *assegna da*.

La funzione `dblib:rowexec()` da aggiungere alla parte 3 della libreria di base `dblib.lua`, è molto semplice ed è la seguente:

```
function db:rowexec(query)
    local conn = self.conn
    return conn:rowexec(query)
end
```

Il grafico può quindi essere composto con questo sorgente in cui è sufficiente modificare il valore della macro `\nfiumi` per variare il numero dei fiumi da rappresentare nell'istogramma, e nell'ordine dal più lungo al più corto:

```
% !TeX program = LuaJitLaTeX

\def\dbconnect#1{...}% as before
\def\dbclose{...}

\def\selectrowprint#1<=#2;#3{%
\directlua{
local dblib = require "dblib"
#1 = dblib:rowexec([[#2]])
tex.print(#3)
}}

\documentclass[margin=2pt]{standalone}
\usepackage{luatex85}
\usepackage{pgfplots}
\pgfplotsset{compat=1.14}

\newcommand\nfiumi{20}

\newcommand\symbcoord[1]{%
\selectrowprint list <=
SELECT group_concat(nome, ",") FROM (
    SELECT nome FROM fiumi
    ORDER BY lunghezza ASC, nome ASC
    LIMIT #1 OFFSET (
    SELECT count(id) FROM fiumi) - #1);
{list}}

\newcommand\coords[1]{
\selectrowprint list <=
SELECT group_concat(coord , " ") FROM (
SELECT '('|| lunghezza || ',' || nome ||')'
AS coord FROM fiumi
ORDER BY lunghezza DESC, nome DESC
LIMIT #1
);{list}}

\dbconnect{fiumi.sqlite}

\begin{document}
\begin{tikzpicture}
\begin{axis}[xbar,
ymax/.expanded={
\selectrowprint last <=
    SELECT nome FROM fiumi
    ORDER BY lunghezza DESC
    LIMIT 1; {last}
},
ymin/.expanded={
\selectrowprint first <=
    SELECT nome FROM fiumi
    ORDER BY lunghezza ASC
    LIMIT 1 OFFSET
    SELECT count(id) FROM fiumi) - \nfiumi
;{first}},
bar width=9pt,
width=16cm,
height=11cm,
enlarge y limits=0.035,
xmajorgrids,
title={Lunghezze dei principali fiumi
    italiani},
xlabel={km},
symbolic y coords/.expanded={\symbcoord
    \nfiumi},
ytick=data,
nodes near coords,]
\addplot[draw=black,fill=blue!45!green]
   coordinates {\coords\nfiumi};
\end{axis}
\end{tikzpicture}
\dbclose
\end{document}
```

8 Chinook

In questa sezione applicativa dell'articolo metteremo in luce con un ulteriore esempio concreto la tecnologia per la connessione ai database SQLite e l'elaborazione dei dati con LuajitTEX e la libreria LJSQLite3.

Sfrutteremo le macro utente definite alla sezione 6 e in quella successiva per eseguire query e formattarne i dati risultanti, adattando il codice

alla particolare struttura del problema.

Si tratta di produrre documentazione commerciale in cui troveremo insiemi di dati ma anche campi semplici. Per questi ultimi non è conveniente eseguire una query ogni volta che è richiesto un campo. La soluzione che adotteremo sarà quella di eseguire la query una sola volta e rendere disponibili i dati attraverso oggetti Lua in memoria.

Ricordo che in osservanza della regola aurea che trovate enunciata alla sezione 5, la costruzione del database avviene con strumenti e modalità esterne al sistema TEX.

Per questa esercitazione utilizzeremo un database con dati di esempio scaricabile dalla rete Internet, il Chinook Database https://chinookdatabase.codeplex.com/ versione 1.4 rilasciato con licenza MIT. L'archivio contiene dati di clienti, prodotti e fatturazioni, di una plausibile attività commerciale basata sulla distribuzione di musica online a pagamento.

8.1 Requisiti di progetto

Ci proponiamo di fissare all'inizio dello sviluppo dell'applicazione documentale le caratteristiche di progetto generali qui elencate:

- tutte le funzionalità Lua dovranno essere memorizzate in un'unica variabile globale di nome stabilito dall'utente per minimizzare i rischi di collisione con altri pacchetti;
- i dati saranno disponibili tramite espressioni in Lua all'interno di macro utente che non comportino la scrittura di codice SQL, indicizzando campi di tabella in dot notation, o funzioni in colon notation;
- aggiungere nuove funzioni Lua di libreria dovrà essere semplice;
- l'utente dovrà poter controllare esplicitamente la connessione al database e il caricamento dei dati;
- efficienza di esecuzione.

Alcuni di questi requisiti sono già stati soddisfatti nell'esercitazione precedente per i report sui principali fiumi italiani, altri invece no. Per essi si dovrà realizzare una nuova struttura con un pacchetto Lua di livello base e un pacchetto LATEX di livello utente.

8.2 Primo report: scheda cliente

Un report come una scheda cliente è adatto per iniziare a produrre qualche idea per il progetto di questa applicazione documentale. Organizziamo i file in modo che ogni report sia salvato in una sub-directory dedicata, all'interno di una directory di progetto dove si troverà il file del database.

Scarichiamo il file binario del database SQLite dall'indirizzo https://chinookdatabase.codeplex.com/downloads/get/557773[8] e, nella sottocartella nominata `schede-cliente`, prepariamo un file per LuajitLATEX.

Siamo pronti per scrivere il sorgente del report minimo che immaginiamo riguardi il cliente numero 28:

```
% !TeX program = LuajitLaTeX
\documentclass{article}
\usepackage{polyglossia}
\setmainlanguage{italian}

\usepackage[dataset=ck]{chinook}
\directlua{ck:connect()}
\directlua{ck:loadCustomer(28)}
\directlua{ck:close()}

\begin{document}
Il nome del cliente è:
\textbf{%
\print{ck.Customer.FirstName}
\print{ck.Customer.LastName}}.
\end{document}
```

Se tutto sarà corretto nel documento compilato vedremo aprendolo questo testo:

> Il nome del cliente è: **Julia Barnett**.

Il pacchetto chinook caricherà la libreria base in Lua nella tabella con il nome indicato dall'utente con l'opzione `dataset`, e definirà le macro utente. La funzione Lua `loadCustomer()` eseguirà la query opportuna verso il database creando l'oggetto `Customer` all'interno sempre dell'unica tabella del dataset.

Nel corpo del documento utilizziamo la macro `\print`, già definita alla sezione 6.1, per recuperare le informazioni e stamparle nel documento. L'argomento richiesto da `\print` è un'espressione Lua valida. Nell'esempio minimo questo argomento è semplicemente il nome della variabile Lua che corrisponderà al nome e al cognome del cliente 28.

Stesso risultato per la stampa del nome lo si può ottenere anche scrivendo un unico comando `\print` tramite la concatenazione di stringhe:

```
Il nome del cliente è: \textbf{\print{
ck.Customer.FirstName .. " " ..
ck.Customer.LastName
}}.
```

Cominciamo ora a occuparci del pacchetto chinook scrivendo quindi il codice nella modalità *top-down*: prima le funzioni di alto livello e poi quelle di base che hanno a che fare con l'SQL.

Creiamo un file di nome `chinook.sty` nella subdirectory del report con il codice:

```
\ProvidesPackage{chinook}
  [2016/12/29 v0.1 User macro for dbnook]
```

8. Se accedete con un browser fate attenzione a scaricare il file compresso per SQLite e non quello preparato per uno degli altri DBMS.

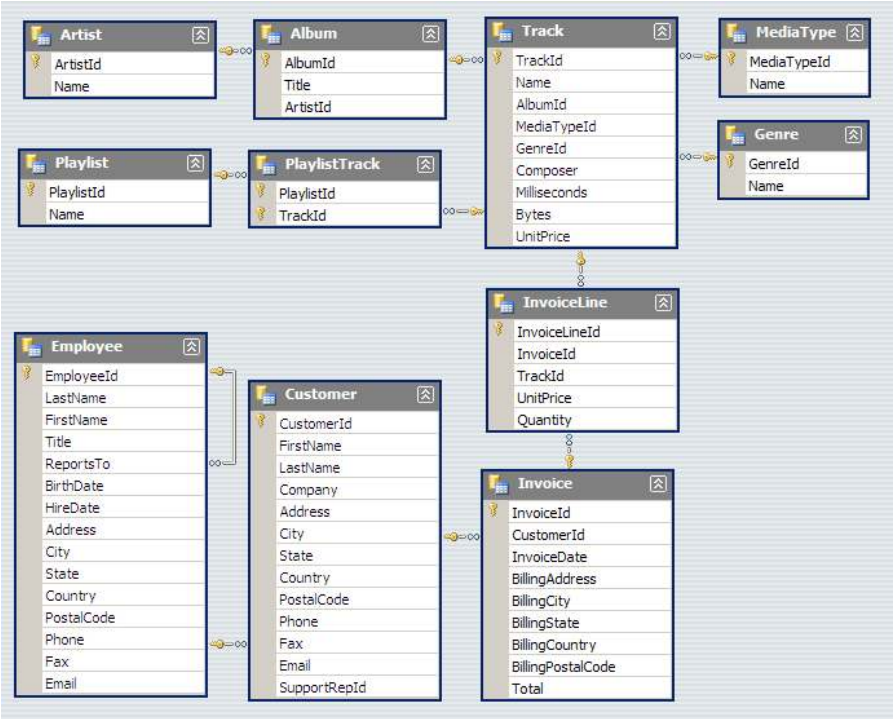


FIGURA 4: Rappresentazione grafica dello schema del database Chinook contenente dati di esempio di un'attività commerciale dimostrativa, e utilizzato nel testo come sorgente dati. L'immagine proviene dalle pagine di documentazione del progetto all'indirizzo `https://chinookdatabase.codeplex.com/wikipage?title=Chinook_Schema&referringTitle=Documentation`. L'uso dell'immagine e del database stesso sono concessi con licenza MIT.

```
\RequirePackage{kvoptions}
\SetupKeyvalOptions{
    family=dbnook,
    prefix=dbnook@
}
\DeclareStringOption{dataset}
\ProcessKeyvalOptions*
\directlua{
\dbnook@dataset = require "dbnook"
}
%
\newcommand\print[1]{%
\directlua{
local expr = #1
if not expr then
    error("Espressione non valida")
else
    tex.print(expr)
end
}}
\endinput
```

Sfruttiamo il pacchetto kvoptions per leggere nel formato chiave/valore l'opzione `dataset`. Carichiamo dunque la libreria che ho fantasiosamente chiamato `dbnook` assegnandone il riferimento al nome di variabile indicato dall'utente per espansione della macro `\dbnook@dataset` creata da kvoptions.

La macro utente `\print` è leggermente più complicata di quella incontrata nella sezione introduttiva 6.1. Infatti prima di stampare l'espressione solleva un errore se eventualmente essa valesse `nil`.

La funzione di libreria `loadCustomer()` dovrà quindi provvedere a creare la tabella `Customer` e a caricarvi l'anagrafica. Infatti nel file `dbnook.lua` troveremo il codice:

```
local sql = require "ljsqlite3"
local dbpath = "../Chinook_Sqlite.sqlite"
local tlib = {}

function tlib:connect()
    self.conn = sql.open(dbpath, "ro")
end

function tlib:close()
    self.conn:close()
    self.conn = nil
end

function tlib:loadCustomer(id)
    id = tonumber(id)
    if not id then
        error("The id ".. id ..
        " is not a valid number")
    end
    local query = string.format([[
    SELECT * FROM Customer
    WHERE CustomerId = %d;
    ]], id)
    local conn = self.conn
    local rs, nr = conn:exec(query)
    if nr ~= 1 then
        error("Customer id ".. id ..
              " is not valid")
    end
```

```
    self.Customer = {
        Id        = tonumber(rs[1][1]),
        FirstName = rs[2][1],
        LastName  = rs[3][1],
        Company   = rs[4][1],
        Address   = rs[5][1],
        City      = rs[6][1],
        State     = rs[7][1],
        Country   = rs[8][1],
    }
end
return tlib
```

La query per reperire i dati di un cliente, osservando lo schema del database riportato nella figura 4, dovrà interrogare la tabella `Customer`. In essa oltre ai campi anagrafici, notiamo la presenza della chiave esterna `SupportRepId` relativa all'impiegato responsabile del cliente, verso la tabella `Employee`.

Una query di join come questa dovrebbe essere salvata nel database come vista:

```
CREATE VIEW vw_customer AS
SELECT * FROM Customer, Employee WHERE
Customer.SupportRepId=Employee.EmployeeId
```

Qualche problema di sicurezza ci potrebbe essere nell'esecuzione della query per via dell'*SQL injection*, una tecnica usata per attaccare applicazioni di gestione dati, con la quale vengono inserite delle stringhe malevole di codice SQL all'interno di campi di input in modo che vengano eseguite. Questo codice malevolo dovrebbe essere scritto nell'opzione `Customer` del piccolo pacchetto `chinook` ma nel nostro caso il rischio è praticamente nullo perché siamo noi stessi che digitiamo da tastiera il numero del cliente.

Inoltre dobbiamo considerare che il database è in sola lettura — ricordate la regola aurea? ebbene è utile anche a questo — e che la funzione `loadCustomer()` esegue la trasformazione da stringa a numero e perciò produrrebbe un errore in caso non lo fosse.

Non aiuta invece il codice che ho scritto per la funzione `loadCustomer()` perché cade nella trappola — che abbiamo visto essere abbastanza remota — sostituendo nel codice della query l'argomento direttamente con la funzione Lua `string.format()`. Avrei dovuto invece utilizzare un *prepared statement*, una funzionalità che crea un oggetto controllato prevedendo l'operazione di binding dei parametri.

8.3 Struttura dati

La tabella Lua creata dal pacchetto chinook rappresenta il database relativo. In essa non compaiono campi e funzioni ma bensì ulteriori tabelle che raccolgono campi e funzioni. Queste tabelle rappresentano oggetti interni al database.

Nell'esempio è stato implementato l'unico oggetto `Customer` che contiene solo campi e non funzioni. Nella prossima sezione presenterò un modo per implementare funzioni formato in modo semplice come prescrive la strategia di progetto.

8.4 Funzioni formato

Per rendere semplice l'aggiunta di funzioni si può creare una tabella all'interno della libreria in Lua, che le memorizzi in relazione agli oggetti. Osserviamo il seguente codice:

```
local fnrepository = {
    Customer = {
        FullName = function (self)
            return self.FirstName ..
            " " .. self.LastName:upper()
        end,
        FullAddress = function (self, sep)
            sep = sep or "p"
            local s1, s2
            if sep == "p" then
                s1, s2 = "(", ")"
            elseif sep == "q" then
                s1, s2 = "[", "]"
            else
                error("Option not valid")
            end
            return self.Address .. ", " ..
                   self.City .. ", " ..
                   s1..self.Country..s2
        end,
    },
}
```

si tratta di una tabella contenente la chiave `Customer`, lo stesso nome dell'unico oggetto creato fino a ora, una tabella che a certe chiavi associa un paio di funzioni.

Il linguaggio è normalissimo codice Lua: impiega il costruttore di tabelle con la sintassi di definizione anonima delle funzioni. Se si desidera aggiungerne di nuove, è sufficiente inserirne la definizione per l'oggetto corrispondente considerando che `self` sarà la tabella che contiene l'oggetto stesso.

Al momento della creazione dell'oggetto, queste funzioni verranno aggiunte controllando che non si verifichino conflitti di nome tra chiavi associate a campi dati e a funzioni.

La funzione `loadCustomer()` dovrà essere modificata nella parte finale in questo modo:

```
function tlib:loadCustomer(id)
-- as before
local res = {
    Id        = tonumber(rs[1][1]),
    FirstName = rs[2][1],
    LastName  = rs[3][1],
    Company   = rs[4][1],
    Address   = rs[5][1],
    City      = rs[6][1],
    State     = rs[7][1],
    Country   = rs[8][1],
}
if fnrepository.Customer then
    for fnname, fnref in
        pairs(fnrepository.Customer) do
```

```
        if res[fnname] then
            error(
                "Internal error: "..
                "conflicted key ["..
                fnname ..
                "] for Customer"
            )
        else
            res[fnname] = fnref
        end
    end
end
self.Customer = res
end
```

Qualsiasi funzione aggiunta al repository per l'oggetto verrà automaticamente inserita nell'oggetto stesso. Perché funzioni, l'utente dovrà usare la dot notation per i campi e la colon notation per le funzioni:

```
-- field access
\print{<dataset>.<Object>.<field>}

-- function access
\print{<dataset>.<Object>:<fn>(<ars>)}
```

Il report minimo della scheda anagrafica, fatte le aggiunte di codice riportate in questa sezione nel file `dbnook.lua` potrà essere:

```
% !TeX program = LuajitLaTeX
\documentclass{article}
\usepackage{polyglossia}
\setmainlanguage{italian}

\usepackage[dataset=ck]{chinook}
\directlua{ck:connect()}
\directlua{ck:loadCustomer(28)}
\directlua{ck:close()}

\begin{document}
Il nome del cliente è:
\textbf{\print{ck.Customer:FullName()}}.

Indirizzo:
\print{ck.Customer:FullAddress("q")}.
\end{document}
```

e nel documento PDF potremo leggere:

> Il nome del cliente è: **Julia BARNETT**.
> Indirizzo: 302 S 700 E, Salt Lake City, [USA].

8.5 Documentare le funzionalità

Osservando i tre componenti per creare il report minimo, il file sorgente del report, il codice del pacchetto `chinook` e quello della libreria `dbnook.lua`, possiamo vedere che l'utente deve conoscere sia funzionalità del pacchetto sia quelle della libreria.

Da un lato è necessario conoscere le macro di aiuto come per esempio `\print` e dall'altro è necessario conoscere le funzioni Lua per richiamarle direttamente e i nomi dei campi.

Preparare un file di documentazione non è un lavoro accessorio, e non solo per chiarire come sono distribuite le funzionalità tra TEX e Lua. Sintetici e aggiornati fogli di riferimento consentono agli utenti di risparmiare tempo e fatica.

8.6 Esercitazioni

Un report come una fattura è piuttosto complesso. Presenta un'intestazione con il logo e i recapiti dell'azienda che la emette, una numerazione di identificazione, i dati di anagrafica del cliente e il quadro dei prodotti acquistati che riporta in basso i totali.

Lascio al lettore di realizzare il report in grado di stampare una fattura.

9 Progettazione e implementazione

In questa sezione vorrei riassumere l'esperienza che ho maturato nel creare un paio di applicazioni basate sulla centralizzazione dei dati e la creazione di report con il sistema TEX perché possano essere un utile riferimento per gli altri utenti nell'ambito di attività aziendali o professionali.

Con il termine *applicazione documentale* intenderò l'insieme dei software utilizzati per produrre documenti utili a partire da informazioni disponibili in una sorgente dati. Esempi di possibili applicazioni documentali sono la fatturazione o l'emissione di ricevute, la produzione di documentazione tecnica, la stampa di cataloghi, la stampa di moduli precompilati, la presentazione dei dati in report statistici per l'analisi, le rendicontazioni, le comunicazioni alla clientela, eccetera, nel contesto di piccole imprese di servizi o di singoli settori interni di una realtà aziendale più grande.

Quando la documentazione ha a che fare con un insieme di dati, è solitamente conveniente ricorrere a un database. Potrebbe essere già esistente all'interno dell'organizzazione oppure potrebbe essere creato a partire da database esistenti per l'utilizzo locale e legato a un'unica produzione. Oppure potrebbe essere creato da zero.

L'architettura dell'applicazione è la seguente:

1. un database management system DBMS che offre l'interfaccia SQL verso dati strutturati in uno schema di tabelle e relazioni, viste, vincoli, indici, eccetera;

2. uno o più moduli software con il quale vengono inseriti i dati nel database;

3. uno o più software di reporting compreso dei sorgenti LuaTEX o LuajitTEX.

L'utente è la persona o il gruppo di persone che usano o che desiderano cominciare a usare il sistema TEX come componente del processo documentale. L'obiettivo è quello di implementare una soluzione via via sempre più completa e user friendly acquisendo passo passo le conoscenze per la scrittura e la verifica del codice.

Lo scenario in-house non è interessante perché rende possibile progettare e implementare una soluzione tagliata su misura sui processi, ma lo è perché nel tempo è in grado di mantenere questa caratteristica più facilmente di un modulo software prodotto all'esterno.

Mentre lo svantaggio non è tanto quello di dover acquisire conoscenze informatiche di programmazione e di analisi e modellazione dati, quanto la possibilità che l'utente passi ad altro incarico cessando l'attività di sviluppo e manutenzione dell'applicazione documentale. Ciò non influisce negativamente sui progetti unici circoscritti nel tempo, o nel caso che nell'organizzazione si collabori attivamente condividendo conoscenze anche per mezzo di guide, tutorial interni o anche semplici file `readme.txt` esaurienti e ben scritti.

9.1 (Non) scelta del DBMS

SQLite ovvio! Stiamo pur sempre cominciando a riflettere sulle informazioni mettendole in ordine con qualche linea di codice e non vogliamo dedicarci a configurazioni o a sfogliare pagine di documentazione.

Avremo modo di farlo magari con un database management system come PostgreSQL magari in cloud. In altre parole conta di più raggiungere un qualche risultato in breve tempo sfornando codice SQL eseguito senza errori che realizza un primo database con una manciata di tabelle.

Un prototipo minimo ma funzionante di un report da LuajitTEX vi farà rendere conto del grado di difficoltà del codice Lua e vi darà una sguardo generale sul progetto in sviluppo.

9.2 Costruzione del database

Studiando il processo per costruirne un modello relazionale si ottiene una maggiore consapevolezza sul processo stesso. Si tratta di un lavoro che produce un quadro chiaro e univoco sugli elementi informativi dell'attività che molto probabilmente migliorerà il processo stesso.

Un secondo vantaggio deriva dalla flessibilità con cui il modello si evolve nel senso che è relativamente semplice modificarlo per tener conto di ulteriori informazioni o nuove caratteristiche del processo rappresentato.

Per la mia esperienza la costruzione del modello è in sostanza un'operazione di *scomposizione* di elementi in sottostrutture relazionate tra loro.

Ipotizziamo per esempio di elaborare un modello relazionale per questa stessa rivista con l'obiettivo di gestire il processo di ricezione, accettazione, revisione e pubblicazione degli articoli. Molto semplicemente, si inizierebbe con il rispondere ad alcune domande come: cos'è una rivista? da chi è composta e chi vi contribuisce? come si svolge il processo di produzione?

Individueremo subito la Redazione — che si compone del Direttore, del Comitato Scientifico, e dei Revisori — e gli Autori, e poi il prodotto che è la sequenza di numeri pubblicati periodicamente, ciascuno suddiviso in quattro parti: le pagine di copertina, l'editoriale del Direttore, gli articoli e gli annunci degli eventi d'interesse.

A sua volta, ciascun articolo scritto da uno o più Autori, si compone di diverse sezioni, in genere termina con una bibliografia e appare in diverse versioni, da quella inizialmente inviata alla redazione, a quella pubblicata.

9.3 Costruzione del report

La costruzione di un report comincia con lo scrivere a matita le parti che esso dovrà contenere e prosegue con lo sviluppo del sorgente per gradi, formando il codice TEX delle singole parti dapprima in modo statico e poi aggiungendo le macro che inseriranno nel documento i dati prelevati dal database.

Sarà anche naturale inserire il codice Lua necessario, in file di libreria e non all'interno dei sorgenti TEX del report. Da un lato eviteremo tutti i problemi dovuti all'espansione del testo argomento della macro `\directlua` e causati da simboli che in Lua hanno diverso significato rispetto a TEX, come il carattere percento o il backslash, e dall'altro otterremo una più importante pulizia dei listati.

Sia per il formato LATEX che per ConTEXt dovremo tener naturalmente in conto che il compositore dovrà essere LuaTEX o LuajitTEX perché solo questi programmi sono in grado di caricare per mezzo di Lua librerie esterne per la connessione dati. Per LATEX in particolare non si dovrà caricare il pacchetto inputenc perché la codifica del sorgente è obbligatoriamente Unicode utf8, mentre dovremo sostituire i pacchetti fontenc con fontspec e babel con polyglossia. Per ConTEXt dovremo usare la versione Mark IV.

9.4 Primi passi di sviluppo

Riassumendo, per creare un prototipo minimo per un'applicazione documentale in SQLite e TEX, le cose da fare sono:

1. inserire in un file di testo — chiamato per esempio `build.SQL` — il codice SQL con le definizioni delle tabelle corrispondenti agli oggetti che rappresentano l'informazione reale del processo, ed eseguire il file con il comando da console:

   ```
   $ sqlite3 nomedb.sqlite ".read build.SQL"
   ```

2. popolare il database con dati di esempio tramite un secondo file di testo contenente il codice SQL ed eseguirlo allo stesso modo del passo precedente;

3. creare un report con un sorgente per LuajitTEX che esegua una query e stampi i dati nel documento.

Successivamente si potranno scrivere programmi appositi per reperire i dati e inserirli nel database. Questi moduli sono del tutto indipendenti da quelli che realizzano i report. Per esempio, io ho fino a ora sempre utilizzato il formato *data description* di Lua per inserire i dati strutturati in semplici file di testo, e script dedicati per leggerli e salvarli nel database.

Questo formato non è altro che l'uso della sintassi del costruttore di tabelle di Lua unito alla proprietà che è possibile omettere le parentesi tonde della chiamata di funzione se l'argomento è una stringa o una tabella.

I file dati appaiono come una sorta di formato JSON quando in realtà sono una serie di chiamate a funzioni Lua con un argomento tabella. In uno script basta scrivere la definizione della funzione e caricare il file dati con la funzione `dofile()`. Informazioni dettagliate sul formato che ho chiamato data description sono reperibili nell'articolo (Giacomelli e Pignalberi, 2015) alla sezione 9.

L'interazione con i dati è solamente asincrona all'interno dell'editor di testo, ma è difficile immaginare qualcosa di più semplice e rapido da implementare.

10 Conclusioni

Come mostrano gli esempi concreti e compilabili illustrati in questo lavoro, è possibile comporre un documento TEX con dati prodotti da query verso database relazionali.

I due nuovi motori di composizione LuaTEX e LuajitTEX, essendo Lua-powered, sono in grado di eseguire il caricamento di librerie esterne in grado di istanziare interrogazioni verso i più svariati DBMS tra i quali SQLite, particolarmente adatto per sviluppare applicazioni documentali da utenti e non da sviluppatori professionisti.

Questa tecnologia da un lato produce report di elevata qualità tipografica con viste sui dati delle più svariate tipologie, grafici tabelle, figure e testo naturalmente, e dall'altro costituisce un *sistema* con cui l'utente elabora soluzioni via via più complete e sofisticate, al passo con l'evoluzione delle necessità.

La modalità di creazione del report non è basata su un template, il modello di documento, ma sulla redazione di un normale sorgente TEX in cui si ritrovano macro utente progettate per interagire *direttamente* con il database e formattarne i risultati.

Il punto di vista rispetto alla costruzione di un template cambia perché non si lavora più dall'esterno su un modello ma dall'interno sul documento stesso, unico e particolare. Vale perciò l'idea *un sorgente per ogni report* piuttosto che *un modello per più report*.

Lua si dimostra un ottimo strumento anche per programmare nuovi pacchetti specializzati nel reperire dati relazionali e per integrare moduli software scritti in altri linguaggi.

LuaTEX e LuajitTEX offrono tutta la qualità tipografica dei motori tradizionali del sistema TEX e ancor di più, con la loro attitudine verso i font Open Type, e la possibilità di programmare tutta una serie di sofisticati nuovi strumenti sui dati, particolararmente utili per le applicazioni documentali in azienda o negli studi professionali.

11 Ringraziamenti

Ringrazio Luigi Scarso per il suo lavoro su LuajitTEX e per i suoi suggerimenti sugli argomenti trattati in questo lavoro e Stefano Peluchetti per il suo modulo LJSQLite3.

Come sempre, ringrazio la redazione di ArsTEXnica per tutto il lavoro di revisione fatto sui contenuti che trovate in queste pagine.

Ultimo ma non ultimo, un ringraziamento anche al lettore.

Riferimenti bibliografici

Atzeni, P., Ceri, S., Fraternali, P., Paraboschi, S. e Torlone, R. (2014). *Basi di dati*. MC Graw Hill, 4ª edizione.

Giacomelli, R. (2016a). «LuaTEX + `pdfliteral`». *ArsTEXnica*, (22). URL `http://www.guitex.org/home/numero-22`.

— (2016b). «Primi esperimenti con node di LuaTEX». *ArsTEXnica*, (21). URL `http://www.guitex.org/home/numero-21`.

Giacomelli, R. e Pignalberi, G. (2015). «Generare documenti LATEX con diversi linguaggi di programmazione». *ArsTEXnica*, (20), pp. 40–73. URL `http://www.guitex.org/home/numero-20`.

Ierusalimschy, R. (2013). *Programming in Lua*. Lua.org, 3ª edizione.

Knuth, D. E. (1984). *The TeXbook*. Addison-Wesley Professional, 1ª edizione.

Scarso, L. (2013). «LuajitTEX». *ArsTEXnica*, (15), pp. 59–69. URL `http://www.guitex.org/home/numero-15`.

The LuaTEX development team (2016). *LuaTEX Reference Manual*, 0.95 edizione.

▷ Roberto Giacomelli
Carrara
`giaconet dot mailbox at gmail dot com`

Storia della notazione musicale. Una ricognizione per immagini*

Werner Lemberg

Sommario

La musica è stata ed è tuttora una parte essenziale della vita. Analogamente a quanto accaduto per la scrittura dei testi, anche su come scrivere la musica sono fiorite varie idee. Questo articolo cerca di mostrare, con molte immagini, le soluzioni escogitate lungo più di tremila anni di storia.

Abstract

Music was and is an essential part of life. Similar to writing text there were various ideas how to notate music, and this article tries to show, with many images, the found solutions in the course of more than 3000 years of history.

1 Introduzione

Nel corso dei millenni, l'umanità ha sviluppato molti modi diversi di scrivere la musica. Le soluzioni adottate possono essere classificate approssimativamente come segue.

1. descrizione di azioni
2. parole
3. lettere
4. cifre
5. figure
6. figure stilizzate
7. simboli astratti
8. combinazione delle soluzioni 1-7

Questo ordinamento corrisponde grosso modo anche allo sviluppo storico all'interno delle culture: le più antiche fonti cinesi conosciute sono descrizioni di azioni, la Mesopotamia sembra aver cominciato con lettere e cifre, mentre la moderna notazione occidentale adopera essenzialmente tutto il ventaglio delle possibili combinazioni.

È interessante notare che conservare la musica era di per sé molto meno importante che conservare le parole – la cosa vale per tutte le culture antiche del mondo. Conosciamo numerosi testi di inni e canti, ma la loro musica non sopravvive. Un'altra osservazione è che la gran parte delle culture ha sviluppato unicamente *tablature*. Un'eccezione a questa regola fu un sistema di notazione per i canti in greco antico; tuttavia, con la caduta dell'impero romano tale sistema andò dimenticato. Forse a causa del secolare divieto di introdurre strumenti musicali nella musica sacra del primo cristianesimo, i musicisti dell'Europa occidentale svilupparono nuove modalità per annotare la musica cantata, che alla fine hanno portato alla moderna notazione adoperata oggi in tutto il mondo.

In questa ricognizione, l'attenzione è puntata sulla rappresentazione grafica della musica, ricorrendo a numerose immagini che mostrano sia l'inventiva sia la bellezza delle soluzioni escogitate. Per le descrizioni sarà inevitabile adoperare alcuni tecnicismi musicali; i lettori digiuni di una specifica preparazione in materia, tuttavia, potranno ignorarli e godere delle sole immagini.

Quest'articolo è una versione riveduta e molto ampliata di un lavoro inviato per gli atti del congresso MOTYF 2014.[1] La maggior parte delle immagini qui mostrate è costituita da scansioni ad alta risoluzione; si raccomanda perciò di dare un'occhiata al PDF online in modo da poter ingrandire il documento per i dettagli!

Gli esempi musicali sono stati composti con GNU LilyPond, versione 2.19.43.[2]

2 Mesopotamia – Canti hurriti

La più antica notazione per la musica attualmente conosciuta venne ritrovata nel palazzo reale di Ugarit, un'antica città portuale nella Siria settentrionale, oggi chiamata Ras Shamra (رأس شمرة). Questi cocci di tavoletta d'argilla risalgono al 1400 a.C. circa (figure 3 e 4); la notazione adopera parole e cifre (in accadico cuneiforme) ed è destinata a una lira a nove corde (figura 1).

La tavoletta presenta numerosi problemi, il che rende difficile interpretare i dati in modo significativo.

- Il testo è scritto in un dialetto hurrita poco conosciuto.

*Questo articolo è stato pubblicato su *TUGboat*, 2016, **37**:3, pagine 284–304. È stato tradotto da Tommaso Gordini col permesso dell'autore. Eventuali errori o fraintendimenti dell'articolo originale sono da attribuirsi al traduttore (*N.d.R.*). Desidero ringraziare Anna, senza la quale questo contributo non avrebbe visto la luce (*N.d.T.*).

1. MOTYF. International Students' Moving Type Festival 2014: *Type in Music, the Rhythm of Letters*. Polsko-Japońska Akademia Technik Komputerowych, Varsavia 2015. ISBN 978-83-63103-76-7.
2. http://www.lilypond.org

- Le parole presenti nella notazione rappresentano intervalli, non altezze (figura 5) – si tratta di polifonia? Se così non fosse, qual è l'ordine degli intervalli? Ascendente? Discendente?

- Potrebbero gli intervalli essere riempiti con scale o qualcosa di diverso? In altre parole: è una rappresentazione uno a uno di quanto si dovrebbe suonare o è solo un aiuto mnemonico?

- Che cosa significano i numeri? Ritornelli? Pulsazioni? Qualcosa di completamente diverso?

3 Grecia – La colonna di Sìcilo

L'esempio mostrato in questo paragrafo è la famosa colonna di Sìcilo, vecchia di circa 2000 anni (figure 2, 6 e 7), ritrovata in una delle più grandi città egee dell'antichità, Tralles (Τραλλεῖς, oggi Aydın, Anatolia, Turchia). Si tratta di una lapide raffigurante un epigramma o uno *skolion* (un canto conviviale).

Al contrario dei canti hurriti, l'iscrizione sulla colonna rappresenta il più antico brano musicale conosciuto trascrivibile *quasi esattamente* nella moderna notazione musicale, grazie alle numerose opere scientifiche di Pitagora e di altri che hanno introdotto la notazione musicale per i loro trattati teorici. Purtroppo, il numero di brani musicali che effettivamente adoperano la notazione è molto ristretto; come già detto nell'introduzione, infatti, annotarla non era considerato importante. Inoltre, la conoscenza di questo sistema di notazione è andata perduta nei primi anni del Medioevo; dei canti greci è sopravvissuto solo il testo.

4 Egitto

Non conosciamo alcun sistema di notazione musicale proveniente dall'antica cultura egizia; a quanto pare, la musica era trasmessa esclusivamente per via orale. Gli strumenti musicali sono mostrati in numerose immagini (e alcuni di essi sono addirittura sopravvissuti); è quindi possibile ricostruire almeno la gamma dei possibili suoni, ma nulla di più.

Un documento copto con circoletti colorati, datato tra V e VII secolo d.C., potrebbe essere correlato alla notazione (figura 8); nessuno, tuttavia, lo sa con sicurezza.

5 Estremo oriente

5.1 Cina

Come nell'antica Grecia, anche nell'antica Cina la teoria musicale ha conosciuto un consistente sviluppo. Dal punto di vista musicologico, il più importante sito archeologico è la tomba del marchese Yi di Zeng (曾侯乙墓, Zēng hóu Yǐ mù, nella provincia di Léigǔdūn, 擂鼓墩, Cina), risalente a qualche tempo dopo il 433 d.C. Pietre musicali incise e campane contengono iscrizioni correlate ad altezze, scale e trasporti (figura 9). Tuttavia, non è stata ritrovata alcuna notazione musicale.

La più antica notazione proveniente dalla Cina risale al VII secolo e viene chiamata wénzìpǔ (文字譜), una tablatura scritta per esteso (figure 10 e 11). Si tratta di una descrizione in puro testo su come suonare il gǔqín (古琴), una cetra.

Durante la dinastia Tang (VIII–IX secolo) questo sistema di descrizioni verbali venne notevolmente semplificato e portò alla tablatura jiǎnzìpǔ (減字譜, figura 13). Contemporaneamente, venne inventata un'altra tablatura chiamata gōngchěpǔ (工尺譜, figura 14); entrambi i sistemi adoperano caratteri cinesi, cifre e qualche altro simbolo per indicare le diteggiature.

La moderna notazione non occidentale (簡譜 jiǎnpǔ), introdotta all'inizio del XX secolo, si basa più probabilmente sul sistema francese Galin-Paris-Chevé, pubblicato nel 1818 (figure 12 e 15). Pur non avendo un impatto significativo sul mondo occidentale, è divenuto popolarissimo in Asia poiché è confrontabile con la tablatura gōngchě. Ancora oggi gli spartiti e i canzonieri più tradizionali adoperano la notazione jiǎnpǔ.

5.2 Giappone

Tutti i tradizionali sistemi di notazione in Giappone sono tablature, fortemente influenzati da Cina e Corea, a propria volta influenzata anche dalla Cina (figura 16). Nel corso del tempo, furono sviluppati numerosi sistemi di notazione differenti e specializzati a seconda dello strumento cui erano destinati. Le scuole di musica, in competizione fra loro, hanno raffinato ulteriormente tale specializzazione, fornendo molto selettivamente la conoscenza della pratica strumentale solo ai membri del clan (figura 17). Nonostante questo, però, tutti i sistemi di notazione sono solo dispositivi mnemonici, il che rende impossibile interpretarli correttamente senza un'ulteriore tradizione orale.

5.3 Corea

Analogamente a quanto accadde in Giappone, anche in Corea sia la musica sia la notazione furono notevolmente influenzate dalla Cina.

Nel XV secolo, venne sviluppata la notazione mensurale Jeongganbo (정간보, 井間譜), che forniva i mezzi per specificare con esattezza il ritmo inserendo l'informazione musicale all'interno di una griglia. Questo sistema, il primo in Asia in grado di rappresentare la durata delle note, è adoperato ancora oggi (figura 18); nel XVIII secolo, l'idea di usare una griglia ritmica fu esportata anche in Giappone.

6 India

L'antica India è un'altra delle maggiori civiltà del passato a non aver sviluppato veri e propri sistemi di notazione. Al posto delle note, in genere venivano aggiunti come unici indizi piccoli accenti sopra e sotto il testo. Come previsto, un simile sistema non è riprodu-

cibile senza la tradizione orale da guru (गुरु, maestro) a shishya (शिष्य, allievo).

Le notazioni moderne furono sviluppate da Vishnu Narayan Bhatkhande (विष्णु नारायण भातखंडे, 1860-1936) e Vishnu Digambar Paluskar (विष्णु दिगंबर पलुस्कर, 1872-1931) nell'India settentrionale, principalmente per l'insegnamento della musica e la conservazione delle composizioni tradizionali. Si basano su caratteri e numeri devanagari con una piccola serie di simboli aggiuntivi (figura 21).

7 Medio oriente

Dopo la caduta dell'impero romano, in Medio oriente non venne sviluppato alcun sistema di notazione; per quanto ne sappiamo, esisteva unicamente una tradizione orale. La notazione occidentale venne introdotta a partire dal 1830 circa in Egitto, ma in modo molto limitato.

I teorici musicali Al-Kindi (أبو يعقوب بن إسحاق الكندي, IX secolo) e Al-Farabi (ابو نصر محمد الفارابي, X secolo) usarono le lettere per indicare le corde dell'*oud* (عود, un liuto arabo), insieme alla posizione delle dita. Safi al-Din al-Urmawi (صفى الدين الارموى, XIII secolo) aggiunse le cifre per indicare il ritmo nelle proprie composizioni. Tuttavia, bisogna notare che nessuno di questi sistemi conseguì qualche importanza pratica per suonare la musica.

Probabilmente, il più notevole contributo ai sistemi di notazione del Medio oriente si deve a Dimitrie Cantemir, principe di Moldavia (Kantemiroğlu, in turco), che pubblicò la propria notazione letterale attorno il 1710 pur essendo in esilio forzato a Costantinopoli, raccogliendo e conservando circa 340 brani strumentali ottomani (figure 19 e 20).

8 Europa

8.1 Neumi

Isidoro di Siviglia, vissuto all'inizio del VII secolo, nel proprio libro *Etymologiae* (conosciuto anche come *Origines*) afferma:

> *nisi enim ab homine memoria teneantur soni, pereunt, quia scribi non possunt*[3]
>
> *(infatti, se i suoni non vengono trattenuti dalla memoria dell'uomo, muoiono, perché non possono essere annotati)*

Presto la storia della musica ha fornito controesempi: alla fine del VII secolo, nella Spagna settentrionale cominciarono a svilupparsi i neumi *neumes* visigotici (contrassegni flessivi per annotare la musica) (figure 22 e 23).

Analogamente, i primi neumi paleofranchi comparvero verso l'850 nelle composizioni di Aureliano di Réôme (figura 25).

Nei secoli X e XI, sviluppo e uso dei neumi cominciarono a prosperare in numerose località europee: San Gallo (Svizzera), Laon, Bretagna (Francia), solo per nominarne alcune.

I neumi si possono classificare a grandi linee in *adiastematici* o *diastematici*. I più antichi neumi adiastematici mostrano la direzione di una melodia, ma non le altezze. D'altra parte, ritmo e dinamiche erano abbastanza precisi (figura 26).

I neumi diastematici erano piuttosto il contrario: indicavano altezze abbastanza precise, ma mancavano di indizi su ritmo e dinamica (figura 27).

Probabilmente, i neumi vennero originariamente sviluppati nell'impero bizantino, sulla base di origini greche. La chiesa ortodossa adopera i neumi ancora oggi (con una notazione raffinata).

8.2 Sistemi di linee

Un'altra invenzione fu l'uso di sistemi di linee. Il primo esempio d'uso di linee orizzontali per indicare le altezze si può ritrovare nell'opera teorica *Musica enchiriadis*, scritta nel IX secolo (figura 28).

In seguito, Guido d'Arezzo sviluppò l'idea di un sistema di linee; nel suo libro *Prologus in Antiphonarium* (1030 circa), per indicare le altezze egli raccomandò di adoperare linee a distanza di una terza insieme a una chiave (o a linee colorate).

Con l'introduzione dei neumi quadrati nel XII secolo, lo sviluppo della notazione del canto gregoriano era essenzialmente completata (figura 29) ed è in uso ancora oggi.

Nello stesso secolo cominciò a svilupparsi la polifonia, soprattutto nella scuola di Notre-Dame a Parigi. Essa introdusse anche i *modi* per controllare il ritmo (figure 30 e 31).

8.3 Notazione mensurale

L'invenzione forse più importante nella notazione musicale occidentale si deve a Franco di Colonia (verso il 1250, come è documentato nel suo libro *Ars cantus mensurabilis*). Egli introdusse forme diverse per le teste delle note, a ciascuna delle quali assegnò una durata specifica.

Prima di allora, il ritmo era definito solo implicitamente dal contesto e da regole apprese; la nuova serie di teste di nota permise di annotare durate arbitrarie. Fondamentalmente, questo sistema è ancora lo stesso in uso oggi, solo con piccole modifiche e integrazioni (figure 32, 33 e 24).

8.4 Tablature

Nella musica occidentale, le tablature si diffusero nel XV secolo, principalmente per gli strumenti in grado di produrre più di una nota alla volta. Per indicare chiavi o diteggiatura si usavano sia cifre sia lettere (figure 34, 35, e 36).

8.5 Stampa

La stampa musicale con caratteri mobili cominciò verso il 1500. All'inizio, lo spartito stampato era una copia del manoscritto (figure 37 e 38).

3. Isidorus Hispalensis, Etymologiae, book III, *De Musica*

Stanghette di battuta, legature di valore, legature di frase e altri segni furono introdotti gradualmente nei secoli XVI e XVII (figura 39).

8.6 Incisione

Nel XVIII secolo, la musica polifonica divenne troppo complessa per essere stampata con i caratteri mobili. Al loro posto, per la musica furono introdotte le tecniche di incisione, che cominciarono a essere applicate a lastre di rame (calcografia, figura 41).

Verso il 1730, l'editore musicale inglese John Walsh inventò una nuova tecnica d'incisione: i pentagrammi erano disegnati con uno strumento a cinque punte su una lastra di peltro (una lega costituita principalmente di stagno), i simboli musicali a grandezza fissa vi erano incisi con dei punzoni e ogni altra cosa era realizzata a mano (figura 42).

Nel 1799, Johann André di Offenbach am Main, in Germania, applicò alla musica la tecnica della litografia, di nuova invenzione – le imagini sulle lastre di zinco venivano trasferite meccanicamente (figura 40).

Verso il 1860, l'estetica dell'incisione della musica classica come si usa oggi era completata (figura 43).

9 Il futuro

L'ultimo passo rivoluzionario nella storia della notazione musicale è l'introduzione del computer per comporre la musica – ciò che sta accadendo proprio in questi anni. Oggi il lavoro dell'artigiano incisore alle prese con peltro e punzoni è praticamente estinto.

Tuttavia, fino a poco tempo fa i risultati prodotti dal computer difficilmente potevano competere con gli spartiti incisi a mano. Ciò è dovuto principalmente al fatto che i dati sono bidimensionali, il che rende automaticamente difficile ottenere spartiti piacevoli alla vista. In questo momento la situazione sta cambiando: i computer stanno diventando sempre più potenti, permettendo senza problemi i dispendiosi calcoli matematici necessari per la buona disposizione automatica degli elementi notazionali. Anche il software è in evoluzione: i programmatori stanno imparando dagli errori e dai problemi che affliggono i programmi di prima generazione adoperati per comporre la musica, e forniscono interfacce grafiche migliori con potenti modelli per gli utenti meno consapevoli degli intricati dettagli che interessano un corretto layout musicale.

Fonti

Tutte le immagini non contrassegnate con un URL sono state create dall'autore. Segue una lista con note aggiuntive per alcune immagini selezionate.

1 Immagine estratta da un filmato; mostra Peter Pringle mentre suona un'antica lira.

4 Sul proprio sito Web, Casey Goranson raccoglie non meno di dieci differenti realizzazioni della melodia apparsa in varie pubblicazioni scientifiche, si veda http://individual.utoronto.ca/seadogdriftwood/Hurrian/Website_article_on_Hurrian_Hymn_No._6.html. La maggior parte di esse, se non tutte, sono altamente congetturali per la mancanza di informazioni.

7 La traduzione inglese del testo greco è presa da Wikipedia, https://en.wikipedia.org/wiki/Seikilos_epitaph.

8 La voce del blog http://musicofthebiblerevealed.wordpress.com/2013/07/25/spiritual-harmony-coptic-musical-notation offre maggiori dettagli sulla possibile interpretazione di questa iscrizione. [Il blog non è più raggiungibile, ma l'articolo citato è stato archiviato dalla *Wayback Machine* di *Internet Archive*: https://web-beta.archive.org/web/20140805140101/http://musicofthebiblerevealed.wordpress.com/2013/07/ (*N.d.R.*).]

9 La foto appartiene al blog di Jo Michie in Cina, http://herschelian.wordpress.com/2013/09/25/the-marquis-yi-of-zengs-musical-bells.

11 La trascrizione completa del brano a opera di Thompson si può trovare a http://www.silkqin.com/02qnpu/01yl/transpdf/jsdyl01.pdf.

21 L'immagine è presa dal sito di David Courtney *Music of India*, http://chandrakantha.com/articles/indian_music/lippi.html.

▷ Werner Lemberg
wl at gnu dot org

https://www.youtube.com/watch?v=27opcKxcg1c, per gentile concessione

Figura 1: Lira simile a un *kinnor*.

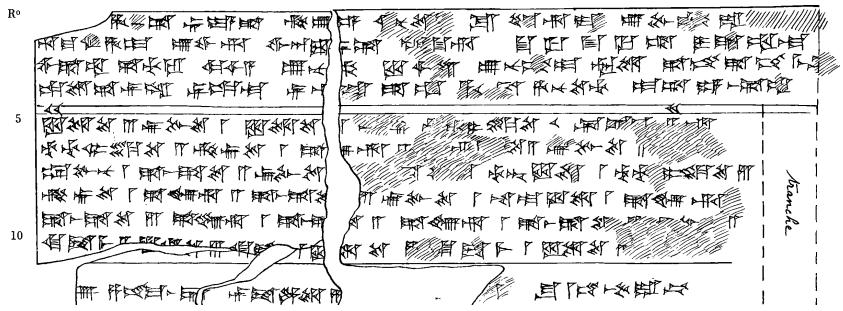
https://commons.wikimedia.org/wiki/File:2._Stèle_portant_l'inscription_de_Seikilos.jpg

Figura 2: La colonna di Sìcilo, Museo Nazionale di Danimarca.

http://www.ramivitale.com/wp-content/uploads/2013/09/H6TabletsFront_lg.jpg

http://www.ramivitale.com/wp-content/uploads/2013/09/H6TabletsBack_lg.jpg

Figura 3: Tavoletta contenente un inno h. 6 (faccia anteriore e posteriore), in forma frammentaria RS 15.30, 15.49, e 17.387 (Museo Nazionale di Damasco).



Rº 5 qáb-li-te 3 ir-bu-te 1 qáb-li-te 3 ša-aḫ-ri 1 i-šar-te 10 uš-ta-ma-a-ri
6 ti-ti-mi-šar-te 2 zi-ir-te 1 ša-aḫ-ri 2 ša-aš-ša-te 2 ir-bu-te 2
7 um-bu-be 1 ša-aš-ša-te 2 ir-bu-te 1[+X] na-ad-qáb-li 1 ti-tar-qáb-li 1 ti-ti-mi-šar-te 4
8 zi-ir-te 1 ša-aḫ-ri 2 ša-aš-ša-te 4 ir-bu-te 1 na-ad-qáb-li 1 ša-aḫ-ri 1
9 ša-aš-ša-te 4 ša-aḫ-ri 1 ša-aš-ša-te 2 ša-aḫ-ri 1 ša-aš-ša-te 2 ir-bu-te 2
10 ki-it-me 2 qáb-li-te 3 ki-it-me 1 qáb-li-te 4 ki-it-me 1 qáb-li-te 2

Figura 4: Trascrizione del testo cuneiforme sotto le due linee di notazione musicale, secondo Manfred Dietrich e Oswald Loretz (*Kollationen zum Musiktext aus Ugarit*, Ugarit-Forschungen 7, 1975).

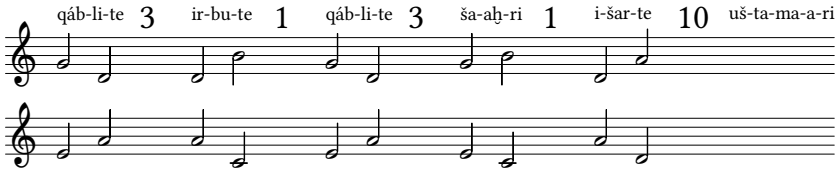
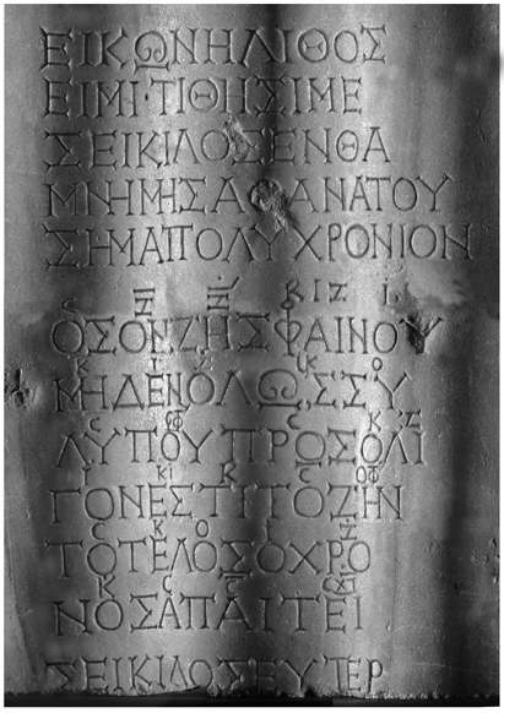


Figura 5: Intervalli e contatori adoperati nella tavoletta dell'inno. A seconda dell'ordine delle corde (ascendente o discendente), che non conosciamo, è corretto il primo o il secondo rigo.


https://commons.wikimedia.org/wiki/File:Seikilos2.tif

ΕΙΚΩΝΗ ΛΙΘΟΣ
ΕΙΜΙ · ΤΙΘΗΣΙ ΜΕ
ΣΕΙΚΙΛΟΣ ΕΝΘΑ
ΜΝΗΜΗΣ ΑΘΑΝΑΤΟΥ
ΣΗΜΑ ΠΟΛΥ ΧΡΟΝΙΟΝ
C Z̄ Z̈ KIZ Ï
ΟΣΟΝ ΖΗΣ ΦΑΙΝΟΥ
K̄ I Ż Ï̄K̄ O
ΜΗΔΕΝ ΟΛΩΣ ΣΥ
C̄ Ö Φ̈ C K Z
ΛΥΠΟΥ · ΠΡΟΣ ΟΛΙ-
Ï K̈Ï K C̄ Ö Φ̈
ΓΟΝ ΕΣΤΙ ΤΟ ΖΗΝ.
C K O Ï Ż
ΤΟ ΤΕΛΟΣ Ο ΧΡΟ-
K̇ C C̄ CẊ Ï
ΝΟΣ ΑΠΑΙΤΕΙ.

Figura 6: Testo della colonna di Sìcilo; a sinistra, un'immagine appiattita, a destra, una versione del testo con l'ortografia moderna. Si noti il segno di 'I' finale: per poter disporre di ulteriori simboli per la notazione musicale, i caratteri greci venivano sia scritti specularmente sia ruotati.

Εἰκὼνή λίθος εἰμί.
Τίθησί με Σείκιλος ἔνθα
μνήμης ἀθανάτου
σῆμα πολυ χρόνιον.

I am a tombstone, an image.
Seikilos placed me here
as an everlasting sign
of deathless remembrance.

Ὅσον ζῇς φαίνου
μηδὲν ὅλως σὺ λυποῦ·
πρὸς ὀλίγον ἐστὶ τὸ ζῆν.
τὸ τέλος ὁ χρόνος ἀπαιτεῖ.

Mentre si vive, lo splendore
non prova alcun dolore
la vita esiste solo per un breve periodo
e il tempo esige il proprio tributo.

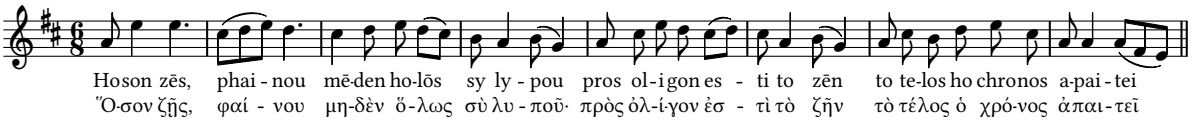
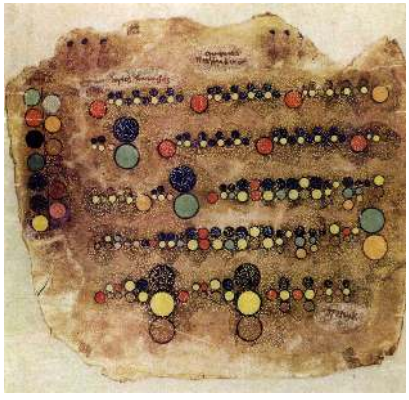


Figura 7: Qui c'è una versione del testo della colonna di Sìcilo in greco con maiuscole e minuscole, insieme a una sua traduzione italiana e a una trascrizione in notazione musicale moderna. La barra sopra una lettera rappresenta due pulsazioni, la barra uncinata tre pulsazioni. Il punto indica una pulsazione non accentata, una legatura sotto le lettere contrassegna una sillaba da cantare in un tempo più lungo di quello impiegato per una singola nota.

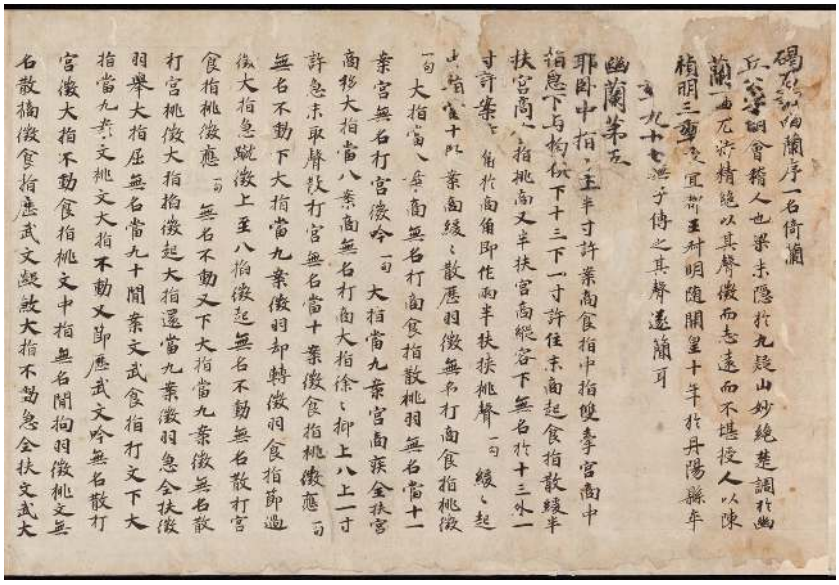


http://musicofthebiblerevealed.files.wordpress.com/2013/07/coptic_musical_notation.jpg

Figura 8: Una delle sei pergamene copte; i colori potrebbero indicare le altezze, il diametro le durate. Metropolitan Museum, New York(?).

http://herschelian.files.wordpress.com/2013/09/bianzhong-concert-2.jpg, with permission

Figura 9: Campane dalla tomba del marchese Yi di Zeng. Museo Provinciale di Hubei, Wuhan.



http://www.emuseum.jp/detail/100229/000/000

Figura 10: L'inizio del lungo rotolo di 4 m con la tablatura per wénzìpǔ del brano *Jiéshí diào yōu lán* (碣石調幽蘭) "Orchidea solitaria, nella forma di una tavoletta di pietra", VII secolo (Museo Nazionale di Tōkyō, TB-1393).

耶臥中指｜下半寸許案商，
食指、中指雙牽宮商。
中指急下与拘倶下十三下一寸許。
住。末商起。食指散緩半扶宮商。
食指挑商。又半扶宮商。
縱容下無名於十三外一寸許案商角，
於商角即作兩半扶、挾挑聲。

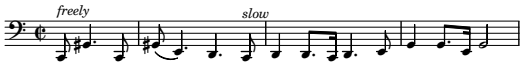


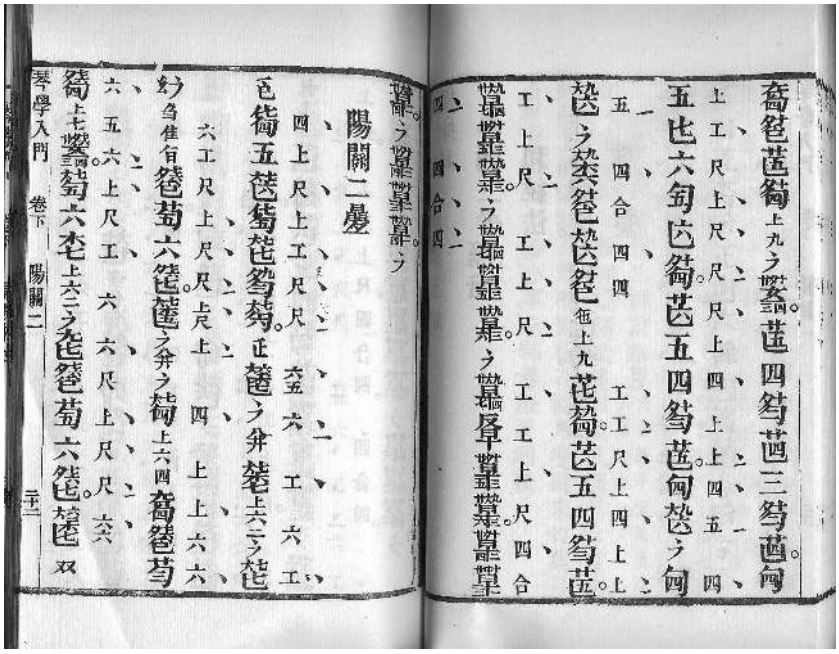
Figura 11: L'inizio di *Jiéshí diào yōu lán*, con la trascrizione di John Thompson. Mentre nell'originale le altezze vengono indicate per i quarti di tono o per intervalli ancora minori, il ritmo può essere solo approssimato.

http://sammlungen.ulb.uni-muenster.de/hd/content/pageview/1474276

Figura 12: Notazione in cifre per il canto *O heil'ge Seelenspeise*, contenuta nel libro *Sursum corda*, un innario cattolico tedesco del 1887. La melodia mostrata si basa su *Innsbruck, ich muß dich lassen (Innsbruck, devo abbandonarti)*, un famoso canto di Heinrich Isaac composto nella seconda metà del XV secolo.

https://en.wikipedia.org/wiki/File:Shenqi_Mipu_vol_3_pg_1.jpg

Figura 13: Due pagine da *Shénqí mìpǔ* (神奇秘譜), risalente al 1425, un esempio di tablatura jiānzìpǔ per cetra qín.



https://en.wikipedia.org/wiki/File:Kam_Hok_Yap_Mun-Yeung_Kwan_Sam_Tip.jpg

Figura 14: Due pagine dallo spartito di *Qín xué rùmén* (琴學入門), risalente 1864, che mostrano la tablatura gōngchěpǔ per cetra gǔqín.

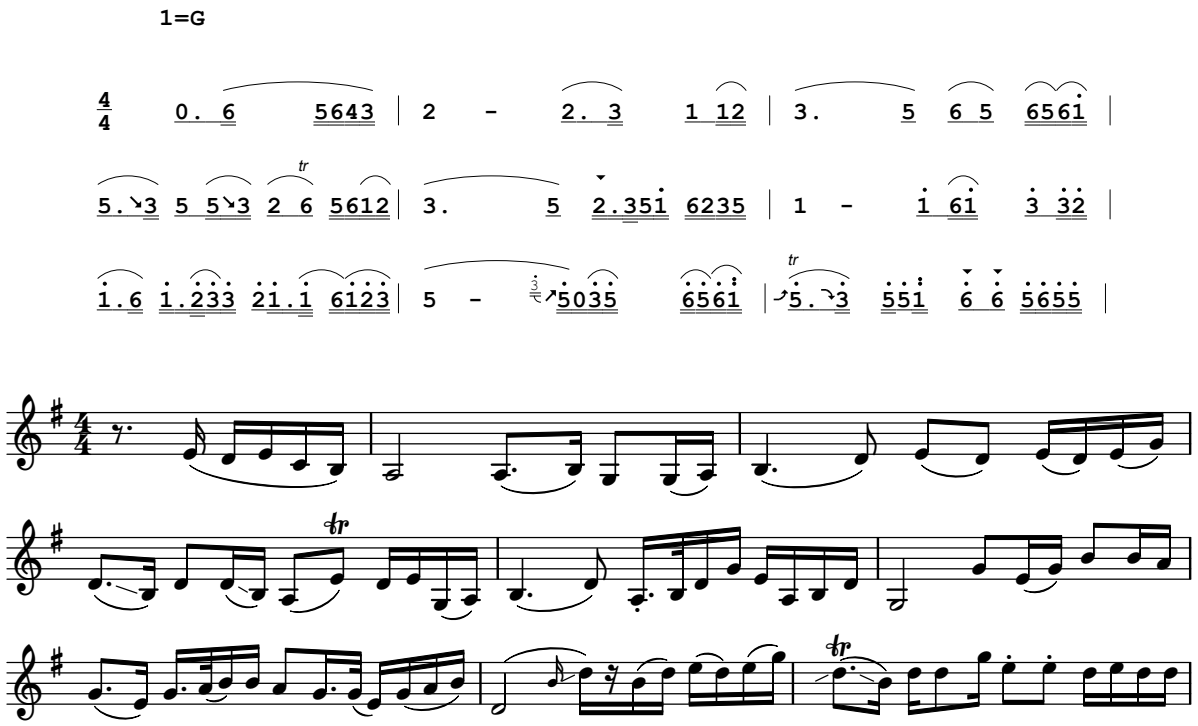
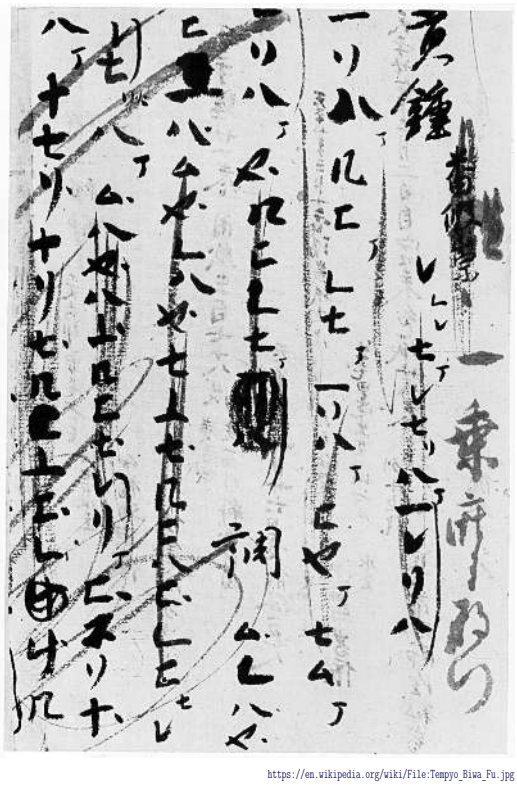


Figura 15: Inizio del brano *La luna specchiata nella seconda primavera* (二泉映月, Èrquán yìngyuè) per èrhú (二胡, un violino cinese); notazione jiănpŭ e occidentale.



https://en.wikipedia.org/wiki/File:Tempyo_Biwa_Fu.jpg

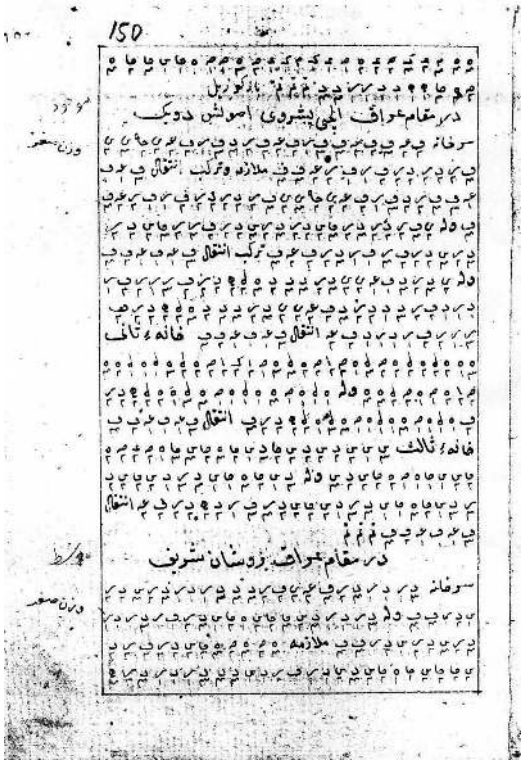
Figura 16: Il *Tempyō biwa fu* (天平琵琶譜, spartito per liuto dell'era Tempyō), risalente al 738 circa. Si tratta essenzialmente di un brano cinese che adopera la notazione per liuto cinese, conservata nel magazzino imperiale (Shōsōin 正倉院) a Nara, Giappone.

http://www.shaku-rus.com/Scores/Ch_score.jpg

Figura 17: Un esempio di tablatura per Shakuhachi (尺八, un flauto da suonarsi a un'estremità).

http://cfile222.uf.daum.net/image/251EC744A5184633226C90A

Figura 18: Un moderno esempio di notazione jeongganbo, da leggersi dall'alto in basso e da destra a sinistra. I riquadri contengono la melodia, i rettangoli sul lato destro le istruzioni sull'esecuzione. Nei riquadri, due righe spezzano la pulsazione in due mezze pulsazioni; due caratteri in una riga spezzano ulteriormente una mezza pulsazione in due quarti di pulsazione. Il segno '–' indica una pausa.



http://www.musikwissenschaft.uni-wuerzburg.de/struktur/lehrstuehle_professuren_ressorts/projekte_und_materialien_des_lehrstuhls_fuer_ethnomusikologie/osman/elci_pesrevi_1_faks/

Figura 19: Pagina del trattato di Cantemir *Kitâb-ı 'İlmü'l-Mûsıkî 'alâ Vechi'l-Hurûfât* (biblioteca del Türkiyat Enstitüsü, İstanbul, Arel 2768).

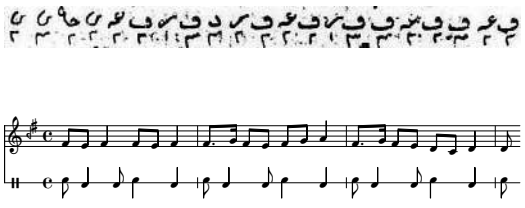


Figura 20: Ingrandimento del quarto rigo della pagina che mostra l'inizio del canto *'Irak Elçi Peşevi', Usul Düyek*, con la trascrizione. La parte inferiore del rigo nel facsimile con le cifre arabe (da leggersi da destra a sinistra) indica il ritmo del tamburo.

http://chandrakantha.com/articles/indian_music/lippi_media/bhat_notation2.jpg, per gentile concessione

Figura 21: Esempio di due sistemi di notazione di Bhatkhande, presi da *Hindustani Sangeet-Paddhati Kramik Pustak Malika* (हिन्दुस्तानी संगीत-पद्धति क्रमिक पुस्तक मालिका), Volume 4. Ciascun sistema consiste di quattro linee.

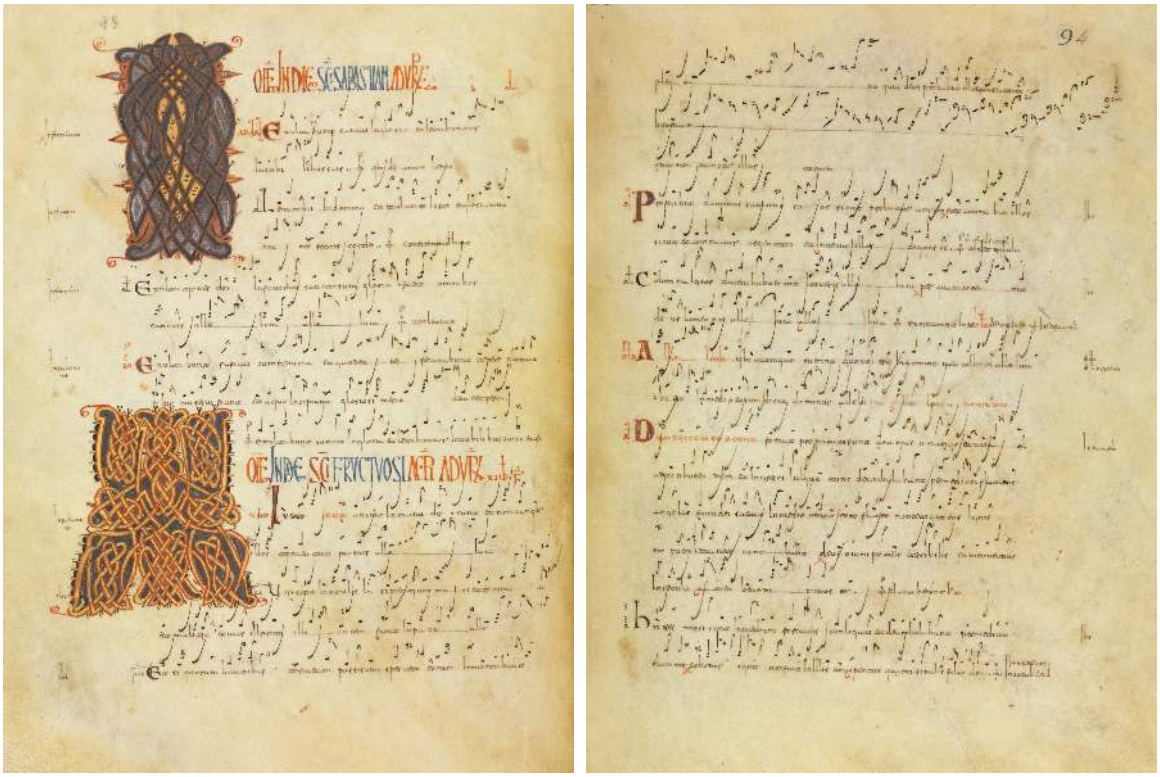


Figura 22: Due pagine dall'*Antifonale visigotico*, molto probabilmente una copia dell'XI secolo di un libro del VII secolo (Archivo de la Catedral de León, Ms. 8). Raffigura il canto mozarabico.

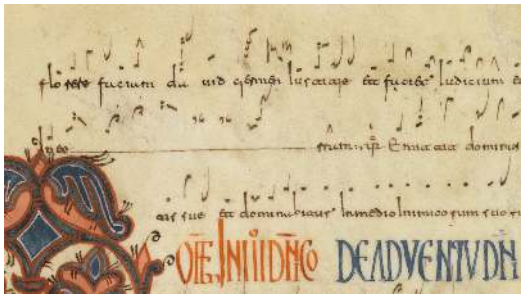
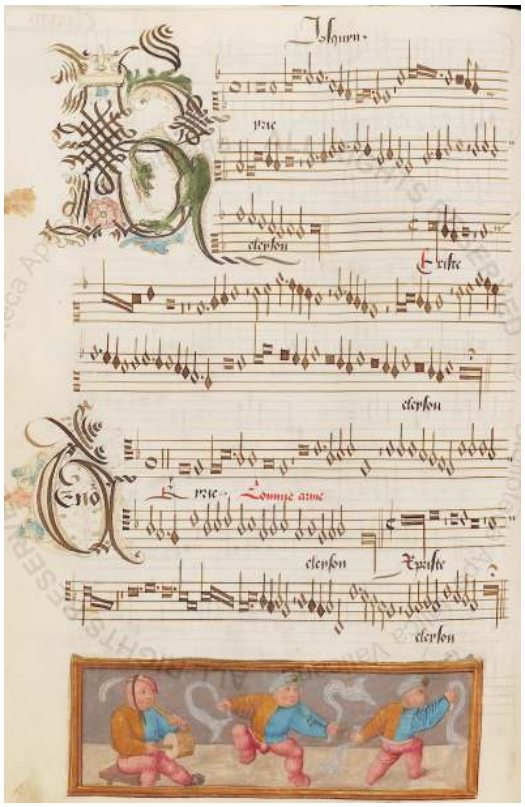
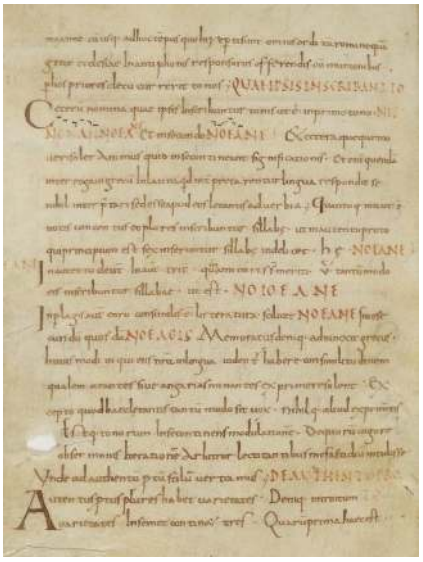


Figura 23: Dettaglio dell'Antifonale, chiamato anche *Antifonario de León*. Oggi questi neumi sono quasi completamente indecifrabili – il rito mozarabico fu interdetto in Spagna verso il 1080 da papa Gregorio VII e sostituito da quello romano.



http://digi.vatlib.it/view/MSS_Chig.C.VIII.234/0384

Figura 24: L'inizio della *Missa L'homme armé* (Kyrie eleison), composta da Josquin des Prez (~1452-1521), in notazione mensurale bianca (Biblioteca Apostolica Vaticana, Chig.C.VIII.234, f. 191v). Sono presenti tutti gli elementi della notazione moderna tranne le stanghette di battuta. L'uso della carta al posto della pergamena ha richiesto di adoperare meno inchiostro per evitare danni. Di qui l'uso di teste di nota cave ('bianche').



http://gallica.bnf.fr/ark:/12148/btv1b8452635b/f147.item

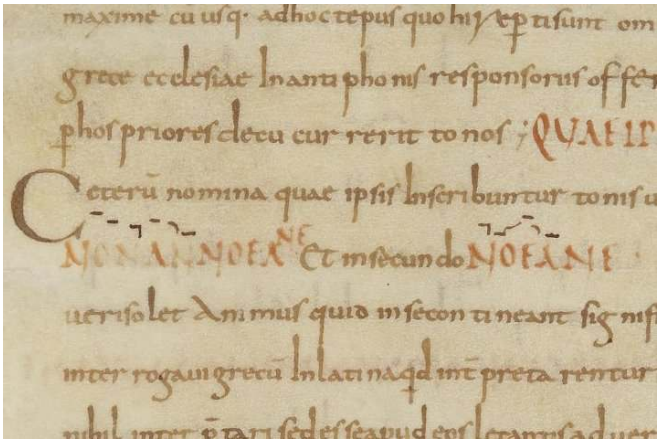
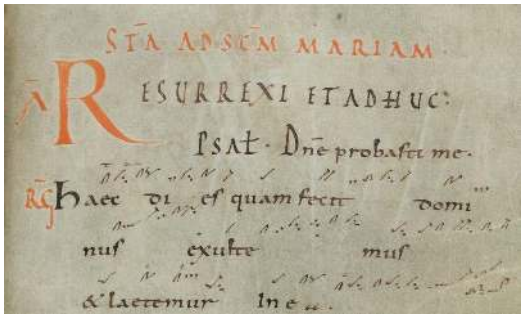
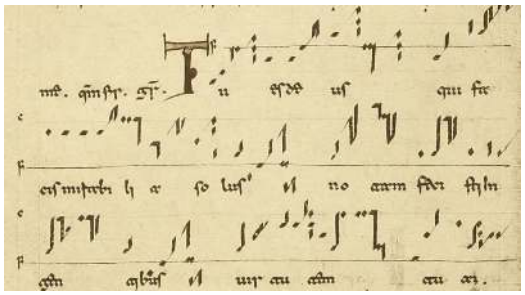


Figura 25: Pagina e particolare del libro di Aureliano di Réôme *Musica disciplina*, un trattato di canto carolingio scritto verso l'850 – contenente solo testo senza musica. I neumi presenti in questa copia dell'880 circa, oggi indecifrabili, vennero aggiunti evidentemente da un antico lettore (Bibliothèque municipale de Valenciennes, Ms. 148).



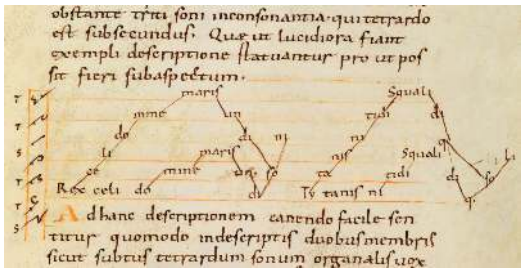
http://www.e-codices.unifr.ch/de/csg/0359/107

Figura 26: Dettaglio della pagina 107 del *Codex Sangallensis 359* manoscritto degli inizi del X secolo, contenente neumi adiastematici (San Gallo, Stiftsbibliothek).



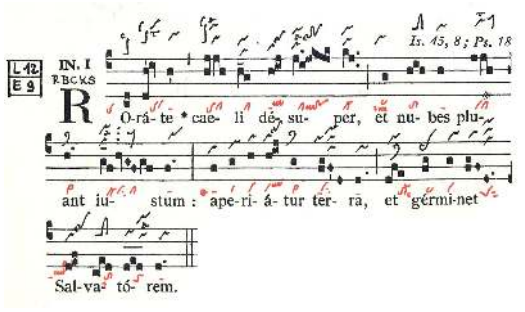
https://archive.org/stream/palographiemusic15macq#page/60/mode/1up

Figura 27: L'inizio del *Tu es deus* nel *Codex Benevento VI.34* manoscritto, f. 59v, scritto attorno al 1100, contenente neumi diastematici (Biblioteca capitolare, Benevento). La linea più grossa fissa l'altezza del 'fa', quella più sottile l'altezza del 'do', una quinta sopra, come indicano le lettere all'inizio delle linee. Alla fine, quelle lettere diventarono le chiavi nella notazione moderna.



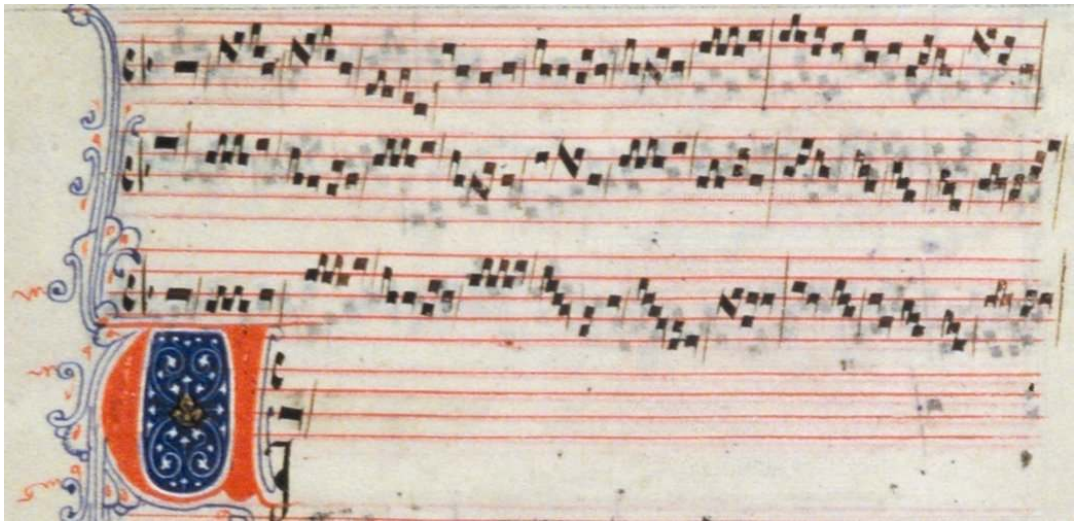
https://commons.wikimedia.org/wiki/File:Musica_enchiriadis_Rex_celi.png

Figura 28: Immagine da *Musica enchiriadis* (Staatsbibliothek Bamberg, Var. 1, fol 57r). Ciascuna linea corrisponde a una corda di uno strumento simile all'arpa.



http://www.kalosconcentus.org/images/Musiche/Intr%20Rorate.jpg, with permission

Figura 29: Il *Graduale Triplex* (pubblicato dall'Abbazia Saint-Pierre de Solesmes nel 1979) mostra sia i neumi diastematici di Metz (sopra, in nero) sia quelli adiastematici di San Gallo (sotto, in rosso), insieme ai neumi quadrati presi dal *Graduale Romanum*.



http://teca.bmlonline.it/ImageViewer/servlet/ImageViewer?idr=TECA0000342136#page/1/mode/1up

Figura 30: L'inizio del *Viderunt Omnes* di Pérotin dal *Magnus liber organi*, composto verso il 1200, che comincia con la sillaba "Vi". Si tratta del primo *quadruplum* conosciuto, un brano per quattro voci diverse (Biblioteca Medicea Laurenziana, Pluteus 29.1, f. 1, Firenze).

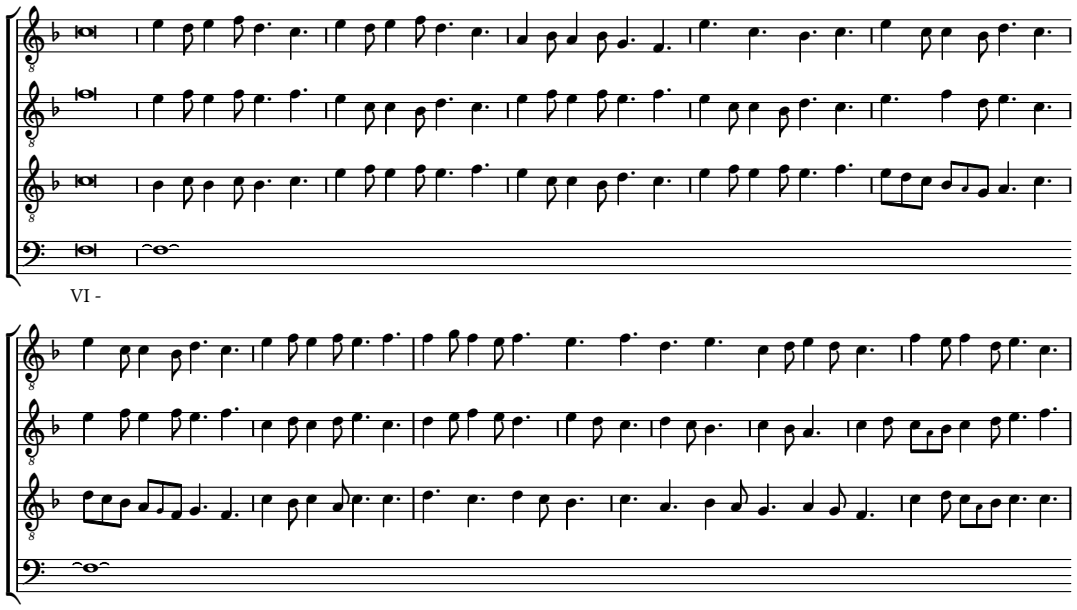


Figura 31: Trascrizione in notazione moderna del *Viderunt omnes* dalla figura 30.

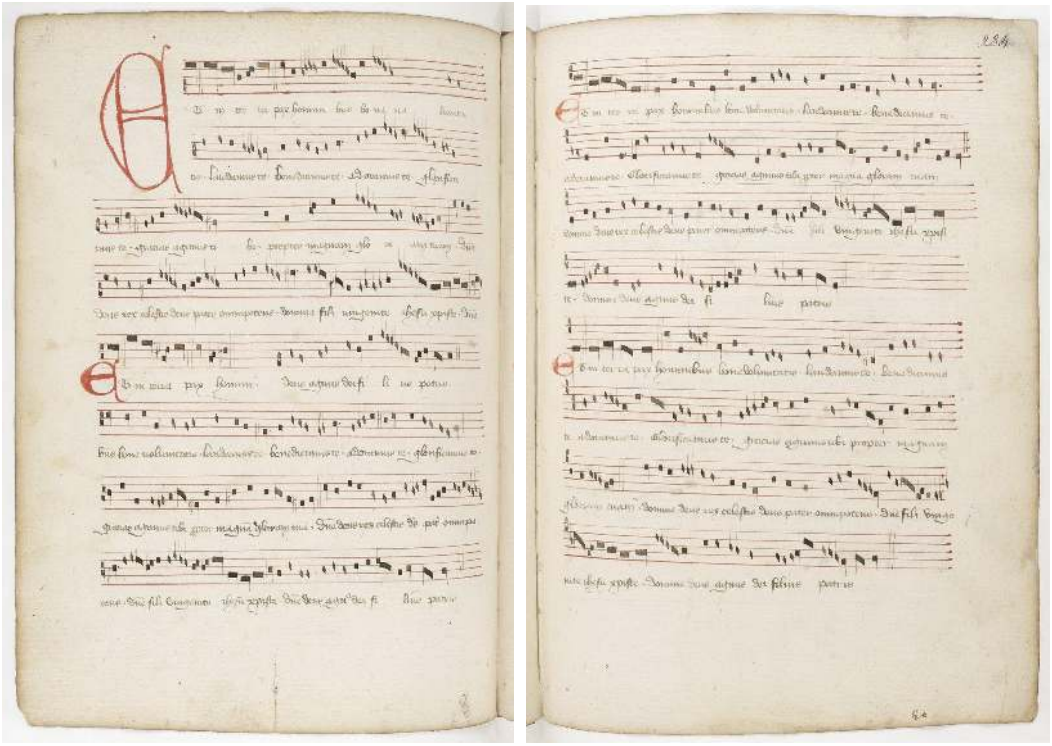


Figura 32: Inizio del *Gloria* della famosa *Messe de Nostre Dame* di Guillaume de Machaut, composta nel 1360 in notazione mensurale nera (ms. Machaut B, f. 283v-284r, Bibliothèque Nationale de France).

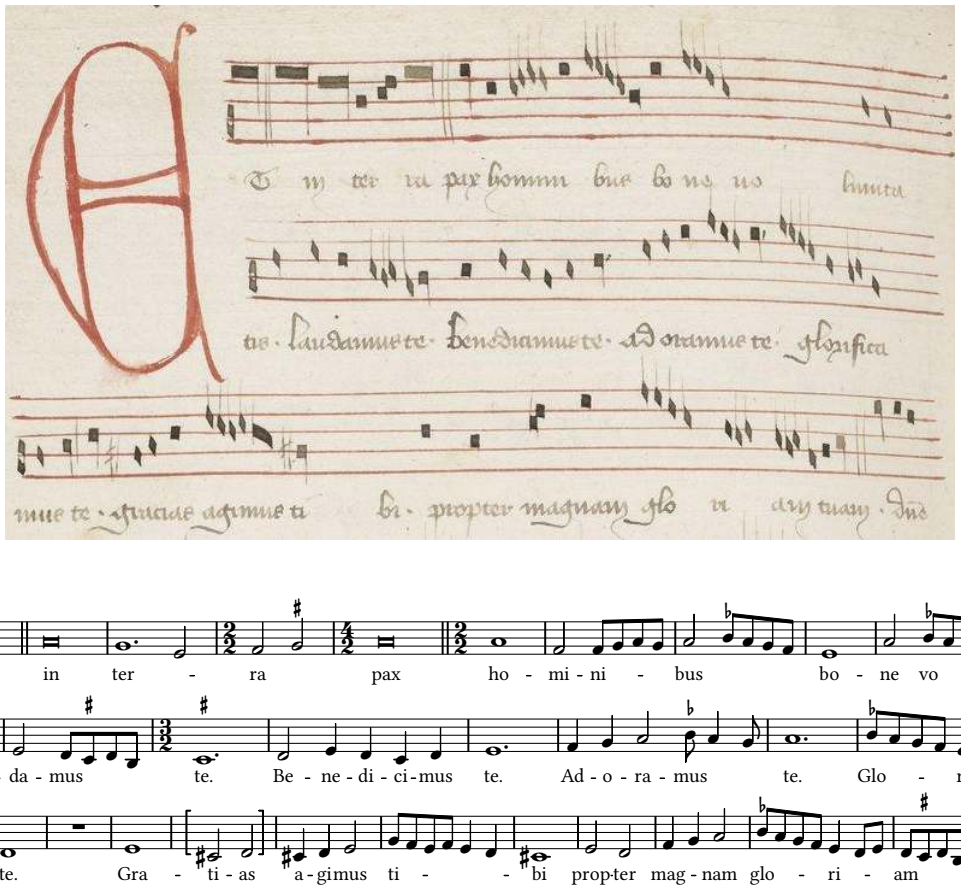


Figura 33: Ingrandimento della voce più acuta del Gloria, insieme a una sua trascrizione in notazione moderna. Poiché il manoscritto originale non contiene indicazioni di tempo, il raggruppamento all'interno delle misure è piuttosto arbitrario.

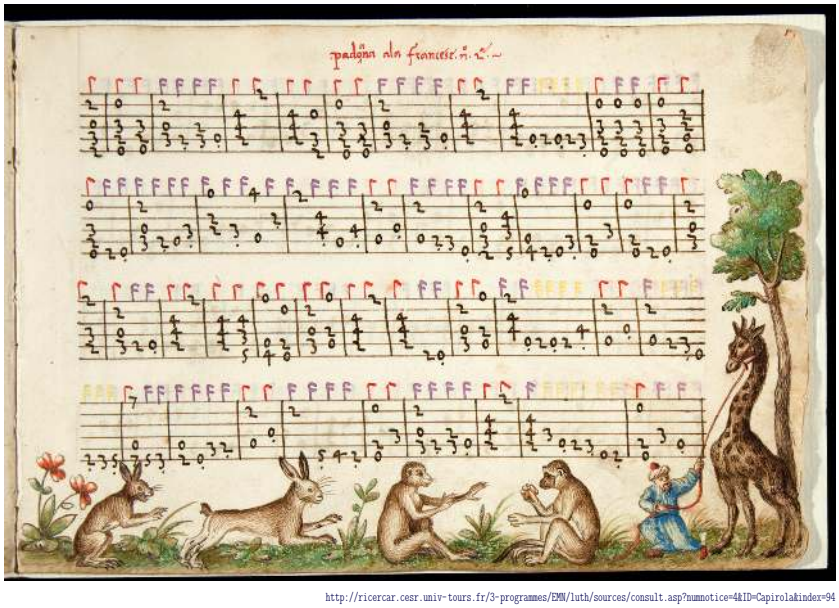


Figura 34: *Padoana a la francese* dal libro per liuto di Vincenzo Capirola, scritto verso il 1517 (Newberry Library, Chicago, MS VM C.25, f. 47r).

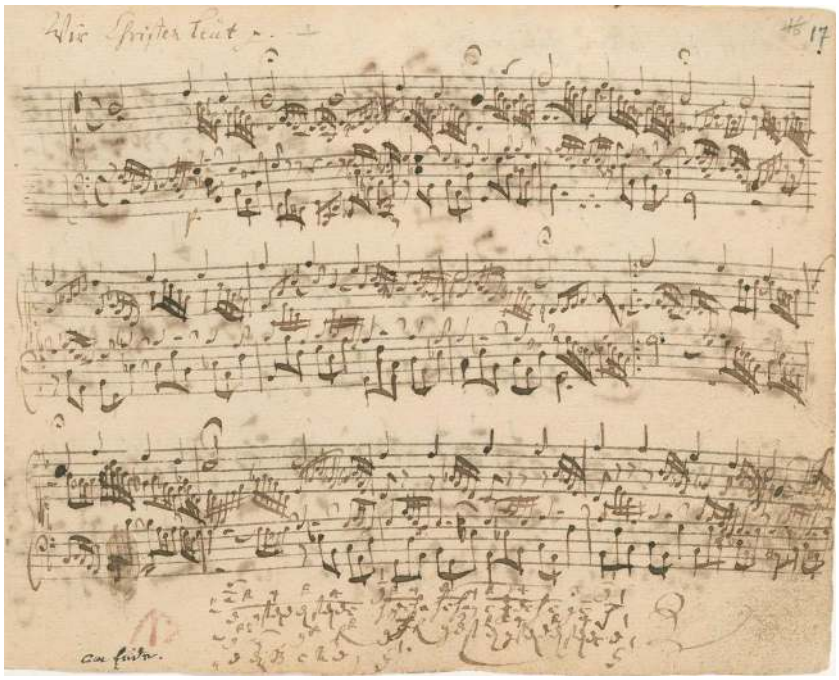


Figura 35: Il corale *Wir Christenleut'*, BWV 612, dall'*Orgelbüchlein* di J. S. Bach, manoscritto, scritto verso il 1715. Le ultime due misure e mezzo sono notate con la tablatura per organo tedesca (Staatsbibliothek zu Berlin Preussischer Kulturbesitz, Mus. ms. autogr. Bach P 283).

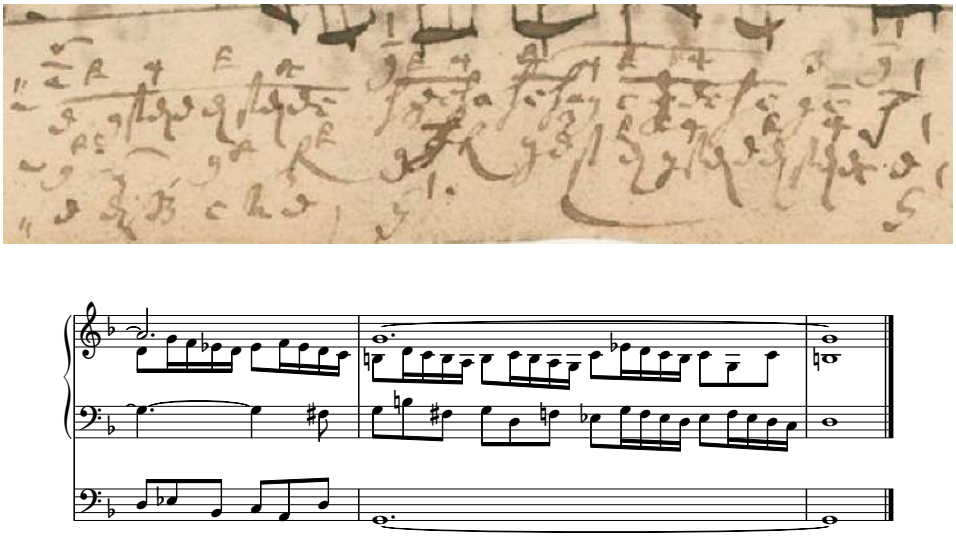


Figura 36: Ingrandimento del BWV 512. Nell'autografo, ciascuna riga con lettere tedesche Kurrentschrift rappresenta una voce; le lettere maiuscole indicano altezze un'ottava sotto, le lettere con una linea sopra altezze un'ottava sopra. Un diesis viene indicato da un tratto curvo e allungato sotto la linea di base. Non vengono usati bemolli, perciò la sequenza di note 'sol-fa-mi bemolle-re' viene notata come 'sol-fa-re diesis-re', per esempio. Le cifre a esponente e gli altri simboli sopra le lettere indicano una durata (cioè '4' per quattro semicrome, '|' per una semibreve).

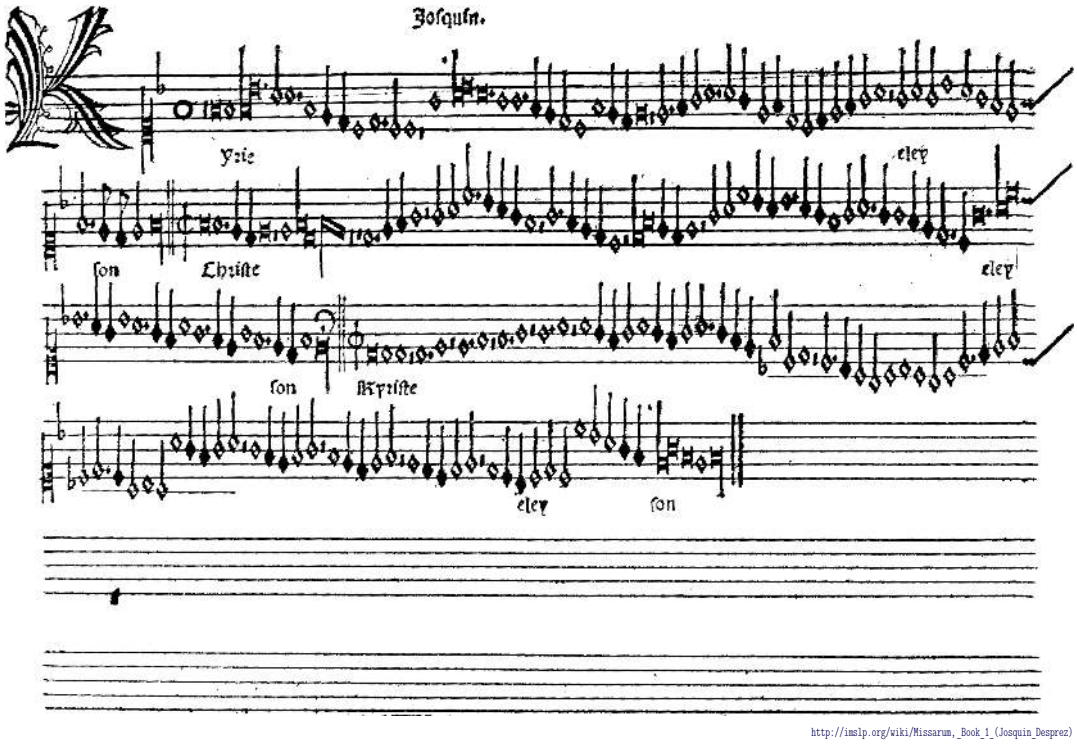


Figura 37: Questa pagina di uno spartito, stampata a Venezia da Ottaviano Petrucci nel 1502, mostra la stessa messa di Josquin della figura 24.

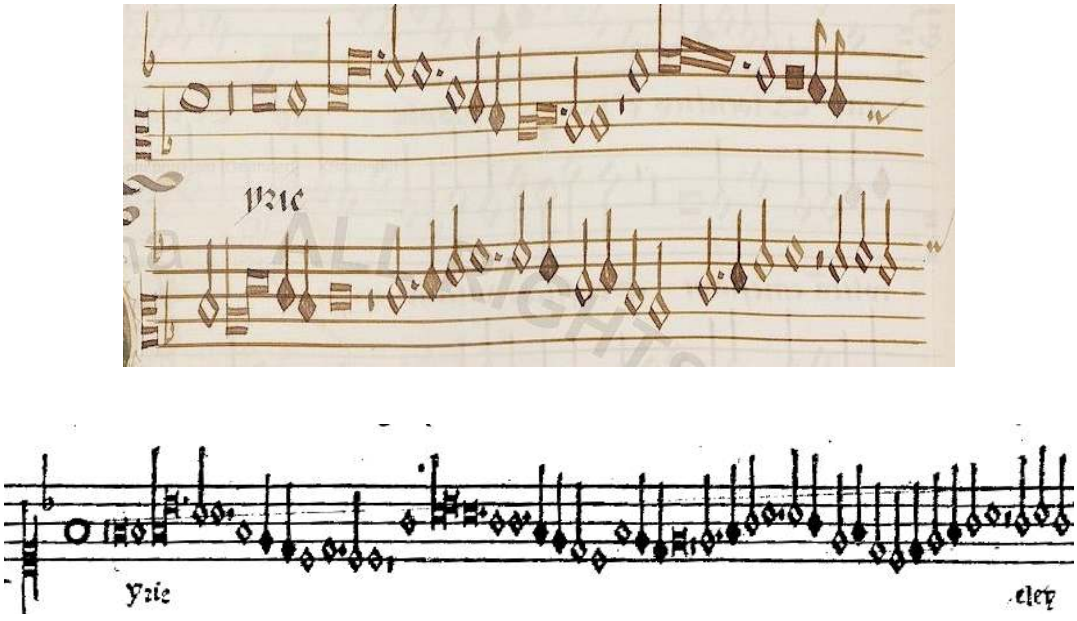
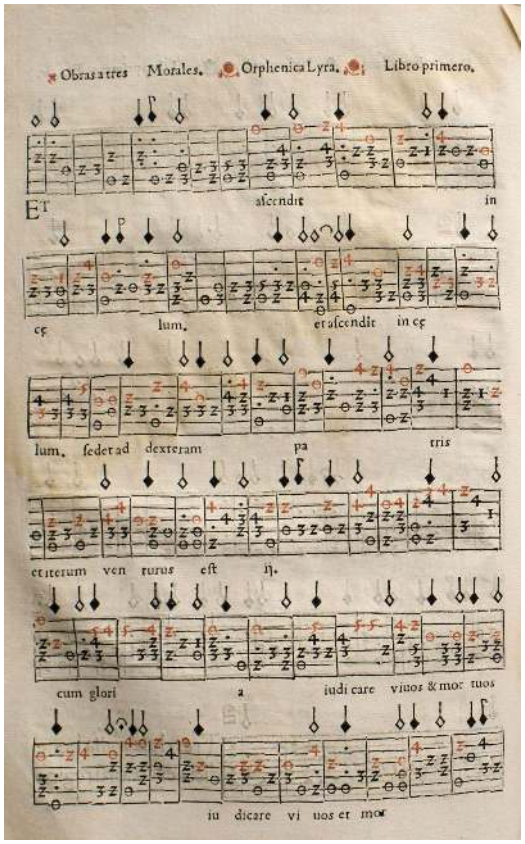
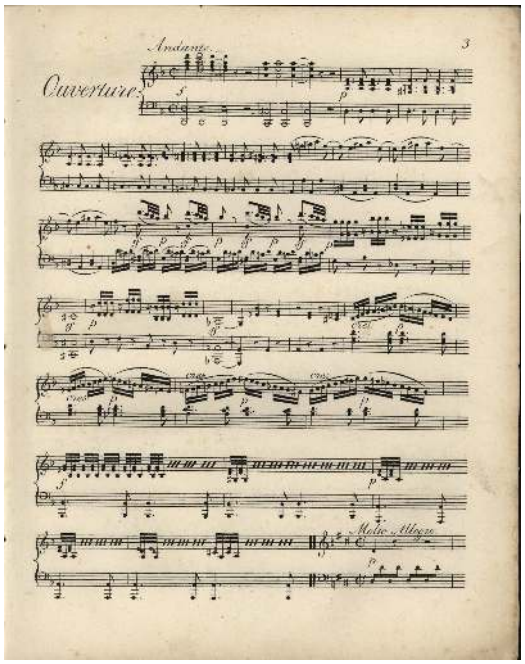


Figura 38: Qui si mostra un confronto diretto tra il manoscritto di Josquin e la sua versione a stampa, nella quale, si noti, viene adoperata una chiave diversa. Contiene anche alcuni errori e varianti (dovuti probabilmente a una diversa copia manoscritta).



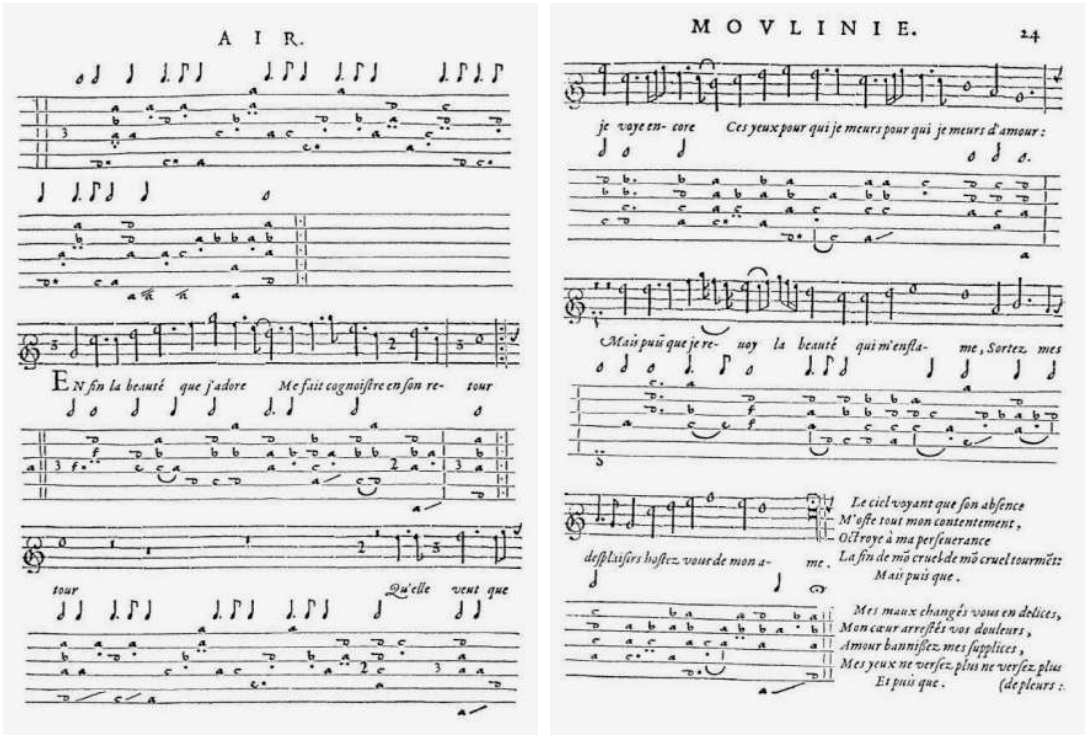
http://www.bibliotecavirtualdeandalucia.es/catalogo/catalogo_imagenes/grupo.cmd?path=1000562&presentacion=pagina&posicion=29

Figura 39: Una pagina dall'*Orphenica lyra* di Miguel de Fuenllana (stampato nel 1554), una tablatura per vihuela (un'antica chitarra), con battute. I numeri rossi indicano la melodia. Viene specificato un indicatore ritmico solo se un ritmo cambia.



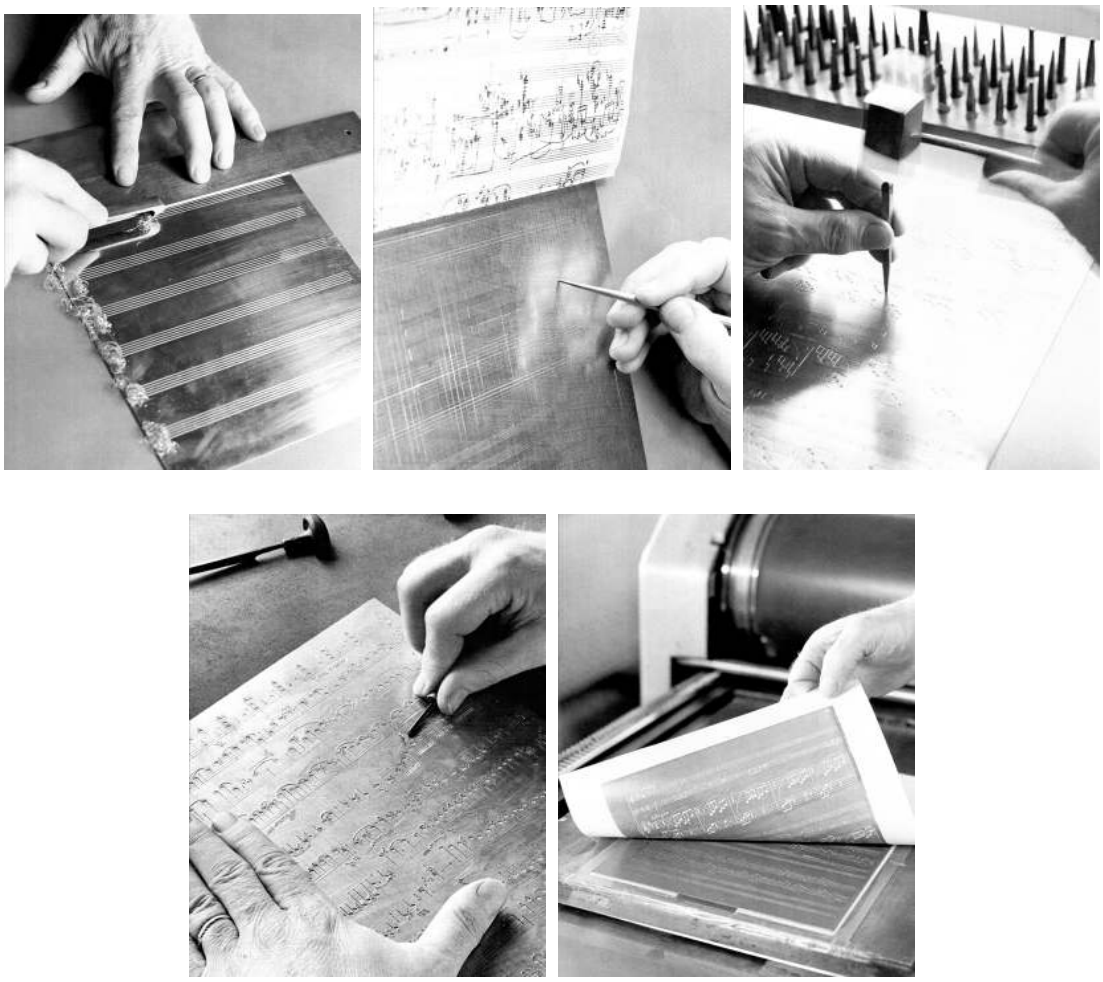
http://www.wurlitzerbruck.com/images/MUS/Mozart%20Don%20Giovanni%20Overture%2010589.jpg

Figura 40: Stampa litografica del 1805 di una riduzione per pianoforte dell'ouverture dal *Don Giovanni* di Mozart, realizzata dalla società di stampa André a Parigi (egli possedeva un brevetto per la litografia).



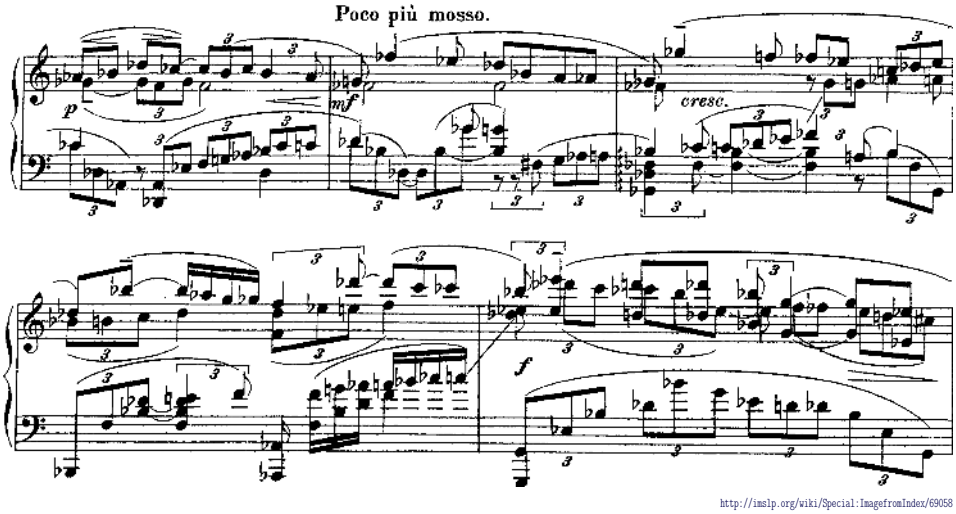
http://imslp.org/wiki/Special:ImagefromIndex/246223

Figura 41: Incisione su lastra di rame dell'aria *En fin la beauté* da Etienne Moulinie, pubblicata nel 1624.



http://www.musicprintinghistory.org/music-engraving/13-about-music-engraving

Figura 42: Il processo d'incisione manuale della musica. Gli spartiti più complessi potevano richiedere anche un giorno intero per completare una sola lastra.



http://imslp.org/wiki/Special:ImagefromIndex/69058

Figura 43: Un campione della seconda sonata per piano di Rachmaninoff inciso nel 1914.

Riviste di aspetto complesso e composte a griglia con LaTeX

Gianluca Pignalberi

Sommario

Il presente articolo illustra alcune soluzioni trovate durante il lavoro di scrittura di una classe per comporre una rivista dall'aspetto abbastanza complesso e composta a griglia, precedentemente realizzata con InDesign.

Abstract

This paper shows some solutions found during the process of writing a class necessary to typeset a journal with a complex layout and grid-typeset, previously made with InDesign.

Introduzione

Nel febbraio 2016 mi è stata commissionata la realizzazione di una classe per comporre una rivista dall'aspetto piuttosto complesso: *Prospettiva Persona*. Molte soluzioni grafiche della rivista si prestavano a essere programmate in TeX; si trattava di una sfida avvincente che mi interessava affrontare e per questo ho accettato il lavoro. Questo presenta ancora qualche aspetto "critico" ma è già in produzione.

L'accordo col committente prevedeva il rilascio della classe su CTAN e la scrittura di un manuale operativo e di un articolo lungo da pubblicare verosimilmente su un numero dedicato di ArsTeXnica. Sto scrivendo — molto lentamente e direttamente in inglese — l'articolo lungo, sto implementando gli ultimi comandi richiesti e sto terminando la correzione degli ultimi bug prima di rilasciare la classe. Nel frattempo mi interessa mostrare alla comunità TeX italiana come LaTeX possa ben vincere delle sfide semplicemente impossibili per prodotti concorrenti.

Dunque questo articolo si limiterà a elencare le sfide vinte per arrivare a produrre la rivista. Consideratelo il *trailer* (o *teaser*, come preferite) dell'articolo che verrà.

1 La commissione

Quando il committente si è rivolto a me per realizzare il lavoro, veniva da una serie di aspettative frustrate: credeva che aggiungendo un po' di pacchetti a memoir e dedicando una decina di ore di lavoro a far funzionare il tutto avrebbe avuto la classe della rivista che componeva con InDesign bella e pronta, con in più la possibilità di avere la bibliografia con la coerenza prima solo sognata grazie ai ben noti strumenti che accompagnano LaTeX. Purtroppo, per ottenere certi risultati, bisogna rassegnarsi alla filosofia del *gong fu*,[1] del duro lavoro, dell'abilità acquisita con fatica. E proprio per continuare a imparare il mio programma d'elezione (LaTeX) ho accettato di automatizzare il più possibile la creazione della rivista di cui vedete alcune pagine d'esempio nelle figure 1–4.

Potete notare l'abbondante presenza di filetti formati da pallini di colore via via più sfumato, di unghie discendenti man mano che viene cambiata rubrica o rivista — *Prospettiva Persona* contiene in sé altre riviste, a cui ci riferiremo come *riviste interne*, come per esempio *Prospettiva Donna* —, note a margine e piedini con l'intervallo di pagine lungo cui si dipana l'articolo corrente, cambî di geometria di pagina e di colonne, immagini con testo sovraimposto, sommario dall'aspetto e dai contenuti articolati.

Consapevole di aver accettato una specie di incubo programmativo, forse non interessante per i grafici ma stimolante per gli informatici, ho iniziato a forgiare e cesellare la classe partendo da report dopo aver fatto una "falsa partenza" con book (che sarebbe andata ugualmente bene in virtù dell'opzione openany).

Ho cercato di tenere un diario di lavorazione il più dettagliato possibile che mi è servito come promemoria sull'attività svolta e su quella ancora da svolgere e mi servirà da guida nella stesura dell'articolo lungo che sarà una specie di "flusso di coscienza" sul processo creativo che ha indirizzato il lavoro.

2 Elenco di produzione

2.1 Composizione a griglia

Come accennato nel titolo, uno dei vincoli di lavorazione era quello di ottenere una composizione a griglia — l'originale aveva qua e là del testo non perfettamente in griglia. La mia storia con tale stile di composizione parte da lontano, da quando intervistai Frank Mittelbach che parlò del suo studio sull'argomento. Poco tempo prima, uno dei commenti di un utente di Free Software Magazine fu proprio relativo alla mancata impaginazione a griglia della rivista "per colpa di LaTeX". Pensan-

1. *Gong fu* è la traslitterazione corretta, secondo i più recenti dettami, del noto termine *kung fu*.

ANNO XXVI
AGOSTO-DICEMBRE
PROSPETTIVA PERSONA
2015/3

Indice

EDITORIALE
C. Goldenstein — A dieci anni dalla morte di Paul Ricœur — 9
A. Giambetti — *Ten years after Paul Ricœur's death*

PENSIERO & PERSONA
M. Schoepflin — Umanesimo cristiano tra ieri e domani — 11
Christian humanism between yesterday and tomorrow

Nelle pagine introduttive dell'opera *Il personalismo*, Emmanuel Mounier indica alcune componenti essenziali della persona e della filosofia che su di essa si fonda. L'essere personale è caratterizzato dalla libertà e dalla comunicazione che lo mette in relazione con gli altri. Il personalismo non è una fredda elaborazione intellettuale, ma un richiamo alla responsabilità e alla testimonianza, secondo uno stile che trova nel cristianesimo il modello più alto.

In introduction of his work The personalism, *Emmanuel Mounier indicates some essential components of the person and of philosophy that is based on it. Personal being is characterized by freedom and by communication that puts him in relation to others. The "personalism" isn't a cold intellectual process, but a recall to responsibility and witness, relating to a style that founds in Christianity its highest model.*

F. Brezzi — Il rinvio ontologico dell'umanesimo di Ricœur — 14
The ontological reference of P. Ricœur's humanism

Il contributo prende le mosse dalla conoscenza diretta del grande maestro personalista e, tramite un'attenta rilettura delle opere (note o meno note) conduce all'orizzonte etico e antropologico-metafisico.

The contribution starts from the direct knowledge of the great personalist master and, through a careful re-reading of the works (more or less known), leads to the ethical and anthropological-metaphysical horizon.

G. Campanini — Condurre la persona al suo sé autentico — 27
Leading the person to its own authentic self

Il personalismo ha una vocazione pedagogica nativa, che contrasta l'enfasi solipsistica dell'io esaltando invece il sé mediante il tu. Se l'individualismo è una sfida, il personalismo è lo strumento indicato.

Personalism has a native pedagogical vocation, contrasting the solipsistic emphasis of the ego by exalting the self through you. If individualism is a challenge, personalism is the indicated tool.

S. Luciano — Carlini fra trascendenza, socialità e storia — 33
Carlini between trascendence, sociality and history

La visione carliniana di persona parte dal pensiero kantiano per affermarne il valore rispetto a qualsiasi unità inglobante: una singolarità che trova il suo compimento in relazione al Trascendente nella mediazione col mondo e con la storia.

Carlinian view of person starts from Kantian thought and aims to affirm its value as compared to any bulky units: a singularity which finds it completion with the "Trascendent" in mediation with the world and history.

STUDI
M. Lusetti — L'accesso alla verità nella *lectio* di Ratisbona — 39
The way for Truth in Regensburg's lecture

A partire dal discorso di Ratisbona (Benedetto XVI) e dalla risposta dialogica dello studioso islamico Aref Ali Nayed, furono discussi i seguenti temi: l'identificazione tra Dio (Suo figlio) e il *Logos*, l'idea greca della ragione e della deellenizzazione (come delineata da Ratzinger). La discussione è impostata per far venir fuori l'accesso storico dell'uomo alla verità trascendente e il valore positivo dell'allargamento della ragione e del fattore religioso, al fine di uno sviluppo armonioso e autenticamente umano della ragione.

Starting from the Regensburg's lecture (Benedict XVI) and from the dialogic response by the Islamic scholar Aref Ali Nayed, the following topics are discussed: the identification between God (His Son) and the Logos, the Greek idea of reason and the dehellenization (as outlined by Ratzinger). The discussion is set to bring out the man's historic access to the transcendent truth and the positive value of the enlargement of the reason and of the religious factor in order to a harmonious and authentically human development of reason.

E. Simonotti — Attraversare la crisi, ripensare l'etica — 45
Time as an anthropology of gift

L'articolo presenta una riflessione sul futuro dell'Europa e sulle sue prospettive etiche. Dialogando con la filosofia di Husserl si sostiene che l'attuale condizione di "crisi" e disorientamento può essere interpretata come opportunità di autentico rinnovamento. Con riferimento a Gadamer viene poi mostrato come il futuro dell'Europa debba essere collegato ad una prospettiva dialogico-ermeneutica e pluralista.

The article presents a reflection on the future of Europe and its ethical perspectives. In dialogue with the philosophy of Husserl it is argued that the current state of "crisis" and disorientation can be interpreted as an opportunity for authentic renewal. Following the philosophy of Gadamer it shows that the future of Europe must be connected to a dialogic-hermeneutic and pluralist perspective.

R.R. De Lorena — Il primato dell'esperienza nello Pseudo-Dionigi — 51
The primacy of experience in the Ps.-Dionysius' works

L'articolo confronta la dottrina mistica dell'Areopagita con gli scritti di Simone Weil e raccoglie notevoli assonanze, soprattutto in ordine allo "svuotamento del soggetto" finalizzato all'esperienza divina.

The article compares the mystical doctrine of the Areopagite with the writings of Simone Weil and collects considerable similarities, especially with regard to the "emptying of the subject" in order to achieve a divine experience.

N. Tosti — Cristina Campo e il suo "Passo d'addio" — 59
Cristina Campo and her "Passo d'addio"

«Passo d'addio sembra si chiami la prova di danza che un'allieva usa offrire staccandosi dalle sue compagne a termine del corso comune. Questo titolo vorrà così suonare quasi il congedo dell'autrice dai maestri e compagni di ieri, alla chiusura di una stagione di esperienze ed esercizi» (Leone Traverso).

*«*Passo d'addio *seems to be called the dance proof that a student uses to offer left her companions at the end of the common course. This title will be intended as the farewell of the author from the masters and the mates of yesterday, at the close of a season of experience and exercises» (Leone Traverso).*

PROSPETTIVA DONNA
G.P. Di Nicola — L'ascolto del "*sensus fidei*" — 67
*Listening to the "*sensus fidei*"*

G.P. Di Nicola — Un'Europa senz'anima e senza madre
A soulless and motherless Europe

La Tarantino s'interroga a livello simbolico sulla svalutazione della funzione materna e del femminile attraverso due filosofe, Maria Zambrano e Simone Weil, che hanno riflettuto profondamente sulla radice violenta della cultura occidentale. La constatazione di un'Europa senz'anima/senza madre esige di recuperare un sapere dell'anima, perché il futuro dell'Europa lo si gioca affrontando le sfide che Simone Weil indicava centrali: la falsa idea di grandezza, la degradazione del senso di giustizia, l'idolatria per il denaro, l'assenza di spiritualità.

Tarantino questions herself at symbolic level about depreciation of maternal role and of femininity through two philosopher, Maria Zambrano and Simone Weil, who deeply examined about violent root of occidental culture. Observation that Europe is soulless/motherless demands to recover knowledge of soul, because EuropE's future is played addressing the challenges that Simone Weil indicated as central: the distort idea of greatness, the degradation of meaning of justice, the idolatry for money, absence of spirituality.

P. Viotto — Dorothy Day, cristiana radicale, radicale cristiana — 71
Dorothy Day, a christian radical, a radical christian

L'occasione di questo contributo è la citazione che, nel suo discorso al Congresso degli Stati Uniti d'America, Papa Francesco ha fatto della poliedrica giornalista e attivista (insieme con quelle di Abraham Lincoln, Martin Luther King e Thomas Merton). Convertita al Vangelo, ha conservato il buono delle istanze femministe e radicali.

The occasion of this contribution is the quote of the multifaceted journalist and activist that, in his speech to the United States Congress, Pope Francis made (together with those of Abraham Lincoln, Martin Luther King and Thomas Merton). Converted to the Gospel, she has kept the good instances feminists and radicals.

Carpaccio, *Storie di Sant'Orsola: la partenza dei fidanzati*, Conegliano (part.)

Gli Autori sono pregati di attenersi alle seguenti *norme redazionali*:

- I testi devono essere inviati per *e-mail* (in un formato *word-processor*, evitare .pdf), corredati dal nome e dall'indirizzo dell'autore, indicati per esteso, e accompagnati dalla qualifica (e dalla città in cui si risiede e/o si esercita la professione).
- La lunghezza complessiva degli articoli non deve superare le 20000 battute (vuoto per pieno) per i contributi da inserire nelle sezioni di *Pensiero e Persona, Studi, Donna* ecc., e le 6000 per le *Rassegne*. Si prega di limitare l'apparato critico a non più di una quindicina di note (essenzialmente bibliografiche).
- I testi devono essere introdotti da un efficace sunto di 500 battute circa, in italiano e in inglese, e i titoli devono essere muniti di occhiello. La responsabilità ultima di titoli, occhielli e *abstract*, è della Redazione, la quale (senza che gli Autori risultino esonerati dall'obbligo di cui è questione) si riserva di modificarli. Eventuali illustrazioni vengano spedite in allegato all'*e-mail*.
- Agli Autori, come segno di gratitudine per la collaborazione all'opera della Rivista, spettano due copie del numero in cui compare il loro contributo. Se vorranno richiederne delle altre, il corrispettivo verrà loro addebitato con uno sconto del 30% (previa dichiarazione del codice fiscale e dell'indirizzo personale).
- Tutti i materiali (foto e testi) inviati alla Redazione non vengono restituiti. Con l'invio, ciascun autore concede la liberatoria sui diritti d'autore (restando naturalmente il legittimo detentore della proprietà intellettuale del testo).

La Direzione declina ogni responsabilità derivante dal contenuto dei singoli scritti, di cui ciascun Autore risponde personalmente.

FIGURA 1: Alcune pagine di *Prospettiva Persona*.

Figura 2: Altre pagine di *Prospettiva Persona*.

Ragione e rivelazione tra Ratzinger/Benedetto XVI e Aref Ali Nayed

L'accesso alla Verità nella *Lectio* di Ratisbona

Mattia Lusetti

I recenti fatti di cronaca, più o meno evidenziati dai media, hanno nuovamente focalizzato l'attenzione sul problema costituito più immediatamente dal rapporto tra religione islamica e violenza, ma più in profondità sul rapporto tra la religione e la libertà e la ragione moderne. Tutt'altro che anacronistica apparirà quindi la ripresa di un breve saggio di uno studioso islamico di origini libiche, Aref Ali Nayed, scritto in risposta alla celeberrima lezione di Benedetto XVI a Ratisbona (2006): saggio che intende essere uno «studio accurato della lezione […] che può predisporre a quel dialogo teologico e filosofico tra studiosi Islamici e Cattolici così urgentemente necessario[1]». Numerosi sono i "fili" delle argomentazioni che costituiscono l'analisi del nostro autore, molti dei quali semplicemente tralascerò, per concentrarmi su due punti focali. Il primo è quello che Nayed riconosce come intento centrale del discorso di Benedetto XVI ovvero «l'importanza di approfondire e allargare la nozione della Ragione Occidentale in modo da accogliere ed includere il contributo che la religiosità rivelata può offrire». Il secondo è in realtà una sfida che il Nostro consegna a conclusione della sua analisi ovvero di ingagiare una critica delle visuali espresse da Benedetto XVI nel suo discorso riguardanti «ciò che significa essere un essere umano ragionevole». L'altro grande filo che attraversa a tratti tutto il saggio e in maniera più diffusa la sua prima parte è la visione dell'Islam che sembra assunta dal Pontefice bavarese nel suo Discorso. Nayed si diffonde molto su questo tema che gli è, evidentemente, molto caro e che ritiene particolarmente esiziale nel Discorso che analizza. Questo è uno dei "fili" che tralascerò a favore dei punti sopra citati che ritengo, senz'altro assieme agli autori citati, essenziali e decisivi in questa congiuntura storica (e non soltanto in questa). Procederò così: dopo aver richiamato i tre principali snodi dell'analisi filosofico-teologica operata da Nayed passerò, a partire dagli spunti di critica che offrono, ad abbozzare un tentativo di ripresa dialogica alle due sfide lanciate dal pensatore libico.

Il dialogo tra l'Imperatore Manuele II Paleologo e il colto persiano riportato da Benedetto è concentrato sul rapporto tra diffusione della fede e violenza. Il culmine dell'argomentazione è raggiunto quando l'Imperatore afferma che diffondere la fede con la violenza è agire contro ragione, e non agire secondo ragione è contrario alla natura di Dio. L'osservazione di Benedetto secondo cui questa affermazione appare connaturata ad un uomo cresciuto e formato nella filosofia greca mentre non è scontata nell'ambito teologico islamico, è accolta come funzionale ad una logica di contrapposizione. Lasciando da parte questa nota mi preme invece di sottolineare in special modo alcune delle istanze critiche che Nayed fa valere di fronte a questa asserzione. La rilevazione di sostanziale ambiguità dell'espressione "ragione" in un contesto aperto come questo, rischia di essere un riferimento vago e fluttuante incapace di sostenere il proseguio dell'argomentazione di Benedetto XVI. È un'istanza da prendere e riprendere seriamente, specie se unita a quella secondo cui «il parlare della natura di Dio è esso stesso problematico»: è la questione dell'arbitrarismo divino e quindi del rapporto tra l'intelletto e la volontà di Dio e la sua stessa Parola (rivelata o "naturale").

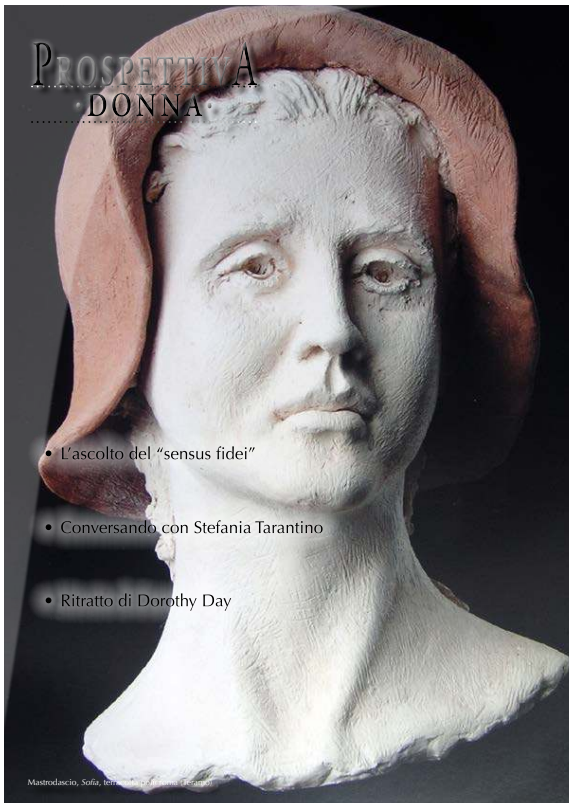
La questione della nozione di ragione conduce al secondo punto ovvero alla constatazione dell'identificazione operata nel Prologo giovanneo tra Dio e il Logos, tra Dio e la Ragione/Parola, autentico fondamento della visione espressa dall'Imperatore bizantino. In Giovanni avverrebbe l'incontro fecondo tra la comprensione di Dio derivante dalla fede biblica e il pensiero greco. A questo punto gioca chiedersi: qual è la ragione a cui Dio è identificato? «È questa la stessa ragione dei Filosofi Greci? Penso di no». Nayed ritiene che la ragione greca sia essenzialmente *theoria*, contemplazione, e non abbia nulla a che fare con la ragione auto-comunicativa e creativa che Benedetto intende trovare (a ragione) in Giovanni identificandola (a torto) con quella del pensiero greco. Il nostro pensatore anzi ritiene che la concezione prevalente nella grecità, opposta a questa, sia quella di una ragione essenzialmente ricettiva di un essere che si auto-comunica. Il Logos giovanneo non può quindi vantare una comunanza con il concetto greco di ragione a parere del Nostro e soltanto un vero e proprio salto concettuale permette di identificarli.

La questione della pertinenza del percorso concettuale delineato da Benedetto del cammino della fede biblica come culminante nel Prologo apre alla terza istanza critica. Per Nayed le affermazioni sul «faticoso e tortuoso cammino» della fede biblica sono nient'altro che una riproposizione rivisitata della filosofia della storia (o della religione) hegeliana. La critica del pensatore libico non si muove tanto dal punto di vista teoretico di ingaggio della fondatezza della visione filosofica di Hegel (e del suo presunto allievo Ratzinger), ma si svolge come critica dell'interpretazione dello sviluppo storico effettivo. Un tale percorso di sintesi concettuale e storico infatti comporta necessariamente una valutazione negativa di ogni stadio antecedente: «mentre la sintesi e il superamento hegeliani suonano meravigliosamente eccitanti a colui con cui si realizza il superamento, è sicuro che vadano ad irritare tutti quelli che risul-

Benedetto XVI durante la Lectio di Regensburg

[1] Cf. Aref Ali Nayed, *A Muslim's Commentary on Benedict XVI's "Faith, Reason and the University: Memories and Reflections"* [accesso 21.11.2015], http://maaad.co.uk/a-muslims-commentary-on-benedict-xvis-faith-reason-and-rhe-university-memories-and-reflections. Le citazioni che seguono saranno tratte d'ora innanzi da questa fonte (traduzione mia). Per il testo di Benedetto XVI cf. Benedetto XVI, *Fede, ragione e università. Ricordi e riflessioni*, Incontro con i rappresentanti della scienza – Discorso del Santo Padre, Aula Magna dell'Università di Regensburg, 12 Settembre 2006, http://w2.vatican.va/content/benedict-xvi/it/speeches/2006/september/documents/hf_ben-xvi_spe_20060912_university-regensburg.html.

PROSPETTIVA -PERSONA- 93-94 (2015/3), 39-44



PROSPETTIVA
DONNA

Via Torre Bruciata 17
64100 Teramo

REDAZIONE
Giulia Paola Di Nicola
Silvia Toma
Anna Vaccarili
Maria Michela Nicolais
Stefania Fuscagni
Maria Laura Di Loreto
Angela Rossi
Cristina Demezzi

Una analogia imperfetta

L'ascolto del *sensus fidei*

Giulia Paola Di Nicola

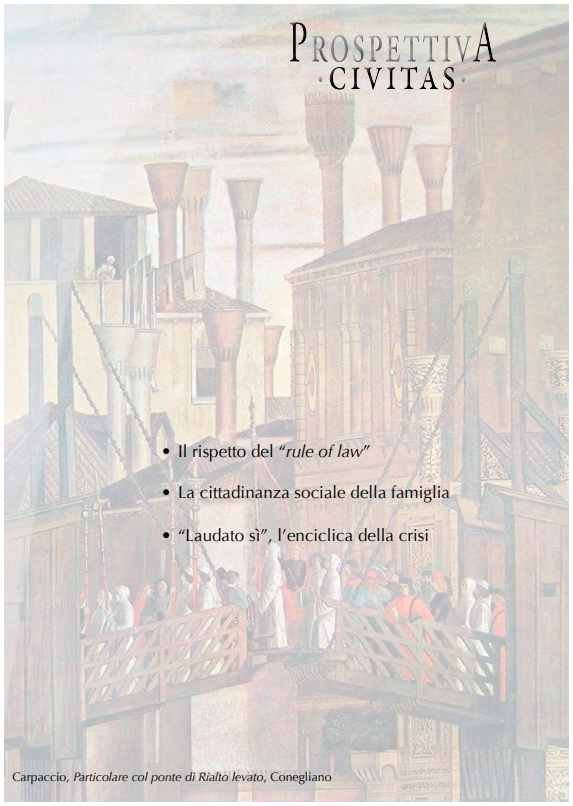
Durante l'Udienza Generale di Mercoledì 6 maggio, Papa Francesco è tornato sui temi della famiglia e ha lanciato delle significative proposte ermeneutiche, nascoste come nel suo stile, sotto un linguaggio semplice e diretto, in piena continuità con il Magistero sulla famiglia di Papa Giovanni Paolo II, ma con una carica rafforzativa e chiarificatrice. Credo vadano sottolineati 4 aspetti:

1. Innanzitutto egli ha esortato non solo i mariti, ma mariti e mogli ad amarsi sull'esempio del Cristo-sposo "senza riserve e senza misura". Nel dire ciò, ha ribadito l'affermazione paolina secondo cui "l'amore tra i coniugi è immagine dell'amore tra Cristo e la Chiesa" (cf. *Ef.* 5, 32), ma sottolineando che i coniugi sono "sottomessi gli uni gli altri" (*Ef.* 5, 21), ovvero reciprocamente a servizio l'uno dell'altro. È dell'amore del resto il mettersi a servizio della felicità dell'altro e dunque "sottomettersi" nel senso di posporre il proprio volere al bene dell'altro. Ciò si trova scritto anche nella *Mulieris Dignitatem*, al n. 24: «L'autore della *Lettera agli Efesini* non vede alcuna contraddizione tra un'esortazione così formulata e la costatazione che "le mogli siano sottomesse ai loro mariti come al Signore; il marito, infatti, è capo della moglie" (5, 22-23). L'autore sa che questa impostazione, tanto profondamente radicata nel costume e nella tradizione religiosa del tempo, deve essere intesa e attuata in un modo nuovo: come una "sottomissione reciproca nel timore di Cristo" (cf. *Ef.* 5, 21)». Eppure una tale convinzione-rilettura della interpretazione più tradizionale di S. Paolo non sembra sufficientemente acquisita nella cultura cattolica, se non addirittura contrastata da talune frange e movimenti (come pure da una letteratura di successo: si pensi al titolo del libro di Costanza Miriano: "*Sposati e sii sottomessa*"), a vantaggio della interpretazione letterale che ingiunge alle mogli di essere "sottomesse". Il tema della reciprocità, contrastato a favore di una complementarietà asimmetrica, è percepito oggi come una bussola orientativa nelle relazioni coniugali, giacché risulta evidente che la raccomandazione unilaterale alle donne è un indiretto appoggio a possibili autoritarismi e soprusi, che inevitabilmente tendono ad emergere in tutte le relazioni umane.

2. Rileggendo l'esortazione ai mariti ad amare le mogli, così come Cristo "ha amato la Chiesa e ha dato sé stesso per lei" (*Ef.* 25), il Papa ha voluto richiamare l'attenzione sull'effetto «enorme – ha detto – che questo radicalismo della dedizione chiesta all'uomo, per l'amore e la dignità della donna, sull'esempio di Cristo, deve avere avuto nella stessa comunità cristiana. Questo seme della novità evangelica, che ristabilisce l'originaria reciprocità della dedizione e del rispetto, è maturato lentamente nella storia, ma alla fine ha prevalso». Nessun dubbio leggendo la precisazione, che Papa Francesco

PROSPETTIVA -PERSONA- 93-94 (2015/3), 64-66

FIGURA 3: Altre pagine di *Prospettiva Persona*.



PROSPETTIVA
CIVITAS

Via Sistina 121
00187 Roma
civitas@prospettivapersona.it

REDAZIONE

Flavio Felice (coordinatore)
Paolo Asolan
Fabio G. Angelini
Mauro Bontempi
Antonio Campati
† Sergio Lariza
Anna Maria Merlini
Maurizio Serio
Fiore Zuccarini

L'inserto "Prospettiva *Civitas*" è stato realizzato con il contributo del Centro Studi Tocqueville-Acton e grazie alla Convenzione con la Fondazione Tercas 2014.

La lezione di Lincoln nelle parole di Papa Francesco

Il rispetto del "rule of law"

Flavio Felice

Dopo l'intervento di Papa Francesco al Congresso degli Stati Uniti d'America e il suo riferimento – tra gli altri – ad Abraham Lincoln, ho pensato che fosse giunto il momento di recuperare nella libreria una vecchia raccolta con alcuni tra i discorsi più celebri del Presidente statunitense. Papa Francesco ha ricordato, in occasione del centocinquantesimo anniversario dell'assassinio di Lincoln, di come questi sia stato "il custode della libertà" e di come abbia "instancabilmente" operato affinché "questa nazione, con la protezione di Dio, potesse avere una nuova nascita di libertà". Così il Pontefice ha potuto riaffermare l'idea, ben consolidata nel Magistero sociale della Chiesa e sviluppata da Papa Benedetto XVI mediante la nozione di "via istituzionale della carità", che, in ordine ai processi di *institution building*, "costruire un futuro di libertà richiede amore per il bene comune e collaborazione in uno spirito di sussidiarietà e solidarietà".

È in questo contesto che il riferimento del Papa al Presidente Lincoln assume un carattere del tutto particolare e forse ancora poco esplorato. Basti pensare che l'enciclopedia teologica post conciliare *Sacramentum Mundi* (1969), con riferimento alla sussidiarietà, riporta di quel principio la seguente prima definizione, proprio di Abraham Lincoln: «l'oggetto legittimo del governo è realizzare quello che una comunità avrebbe dovuto fare, ma che non è stata in grado di fare, o quello che i singoli non possono fare da soli, facendo appello alle proprie capacità. Ma il governo dovrebbe evitare di interferire in tutto quello che la gente può e sa fare da sé». È questo un passaggio significativo, in quanto l'autore, oltre a delimitare i confini dell'azione di governo, impegna la società civile e le comunità che la compongono nel delicato compito di costruire le istituzioni, praticando l'arte dell'autogoverno che necessita di alcune virtù fondamentali.

Tali virtù saranno indicate in un celeberrimo discorso dell'allora giovane avvocato. Era il 27 gennaio del 1838, Lincoln aveva 28 anni e si era da poco trasferito a Springfield, nell'Illinois. La sua carriera politica era agli inizi e, in occasione dell'assemblea di un'associazione giovanile, pronunciò uno dei suoi discorsi più belli e ispirati. Una appassionata difesa del *rule of law*, presentato come il baluardo indispensabile a salvaguardia delle istituzioni politiche, della civile convivenza e del bene comune: "*The Perpetuation of Our Political Institutions*". Il contesto storico di quel discorso era dato da una serie interminabile e orrenda di violenze contro i neri e contro coloro che sostenevano l'abolizione della schiavitù. Lincoln vide in tutta quella violenza una minaccia per la tenuta democratica delle istituzioni americane, del "governo della legge", così come erano state pensate ed edificate dai Padri fondatori, e giunse ad affermare che gli Stati Uniti non avrebbero avuto mai nulla da te-

Recensioni

Filippo D'Andrea, *Pensieri delle piccole cose. Spiritualità dell'essenziale*

Il libro di Filippo D'Andrea, è una raccolta di adagi, aforismi e riflessioni brevi composta tra il 2010 e il 2014; una sezione di poesie, concepite anch'esse come pensieri in versi, chiude il volume. Si tratta di testi che *accesi dal brillio dello scorrere quotidiano* (*Introduzione*, 9), testimoniano un esercizio del pensare cui la parola scritta dà voce attraverso la pagina, e questa voce arriva fino a noi: ci raggiunge sotto forma di intuizioni, di analisi e di sintesi concettuali che nel loro *scrutare dentro gli occhi dei giorni* (*Introduzione*, 11) vi trovano ciò che vi è contenuto, il "senso" delle cose, la cui preziosa, necessaria acquisizione conforta la coscienza e la mente, permettendo di vedere con più chiarezza davanti e intorno a sé.Il D'altronde, proprio il tema del "vedere", concepito come capacità di intendimento e di comprensione, è centrale ai "Pensieri delle piccole cose" e viene enunciato sin dall'affermazione che apre la sezione *Adagi e aforisme*: "Essere capace di vedere dentro la persona è dono spirituale". Tale dono dello Spirito concede, a chi lo riceve, di cogliere la verità esistenziale di cui l'altro è portatore, quella Verità che *si intravede spesso più in uno sguardo che in tante parole*, negli occhi che "sono lo specchio dell'anima", come scriveva Sant'Agostino, ma che si percepisce anche nei silenzi, come suggerisce un adagio della raccolta: "Quando le parole non bastano, ascolta i silenzi". Analogamente, il grande uomo si scorge anche dalle piccole cose, poiché *l'agire interiore determina l'agire pratico*. di conseguenza, un "grazie mancato" o l'impegno nel lavoro che si è chiamati a svolgere possono offrire, rispettivamente, una misura della vuotezza o del valore personale di ciascuno, mentre, come ricorda un aforisma della raccolta, "Il punto più alto della dignità della persona è l'umiltà di conoscere e riconoscere i propri errori, le debolezze, le fragilità" Peraltro, proprio all'umiltà intesa come "laboratorio" al senso del limite sono dedicati diversi dei pensieri contenuti in questa raccolta, che trae il suo nutrimento dalla *spiritualità dell'essenziale*; l'umiltà è il contrario della superbia, tracotante vanità di sé con la quale *troppe volte si invadono sapori sconosciuti*, e trova le sue metafore più belle in due immagini: quella della *punta dei piedi*, la parte più bassa di noi da cui, tuttavia, può partire lo slancio che ci permette di toccare il cielo («Se non tocchi il cielo, alzati sulle punte, la parte di te più bassa»), e quella dello *zero*, valore numerico capace, più di ogni altro, di esprimere non il "nulla assoluto", ma il "tutto" che è in potenza: "Lo zero torna alle origini compiendo totalmente il suo itinerario, infatti è l'unico numero che si completa nel suo cerchio. Lo zero è il numero che consente più di tutti di centrare l'obiettivo. Lo zero è l'unico numero senza spigoli, che possono far male agli altri. Lo zero rappresenta l'umiltà assoluta, via delle beatitudini". D'altra parte, l'umiltà assoluta, che oltre ad essere "via delle beatitudini" è anche assoluta essenzialità e sobrietà, può divenire nondimeno la *leva di una rivoluzione/conversione della coscienza e della vita dell'uomo* che si ponga in viaggio verso il Dio interiore, frontiera del proprio Essere, un Dio che nella prospettiva escatologica della salvezza torna perennemente ad incarnarsi in ogni uomo, poiché, come lasciano sperare le parole di una delle riflessioni brevi contenute nella raccolta, "[...] il nettare del bene, della verità e della bellezza è nel profondo umano". Per raggiungere questi tesori, custoditi dentro di noi, è necessario, con le parole di Giuseppe D'Andrea, compiere "un viaggio dentro le nostre anime e i nostri sentimenti", ma anche intraprendere un cammino "fuori di noi", attraverso *gli eventi dei giorni*, poiché, come rammenta ancora un adagio della raccolta, "Il Natale è avvenuto fuori dalle mura". Si tratta di un cammino faticoso e spesso duro, compiuto a contatto con la realtà, che nella sua crudezza è *laboratorio di santità*, e punteggiato da *tanti incroci* tra i quali *si cerca la direzione ogni giorno*: sapremo di aver raggiunto la meta quando sentiremo che le nostre azioni rispecchiano interamente le nostre intenzioni e noi saremo fieri di assumercene la responsabilità, sia sul piano del vivere personale che su quello del vivere sociale. D'altro canto, il tema della *responsabilità*, come componente di un trinomio inscindibile di cui fanno parte anche *intenzioni* e *azioni*, attraversa in modo forte i pensieri in prosa e in versi che si succedono in questa raccolta: la responsabilità è l'impegno che ciascuno assume sia nei confronti di se stesso, per realizzarsi nell'*orizzonte che si è scelto per la felicità*, sia nei confronti

PROSPETTIVA
-PERSONA-
93-94 (2015/3), 121-128

degli altri, per contribuire alla costruzione di un *tessuto di vita sociale di dignità e di pace*, all'edificazione di un *cammino di speranza*. Tuttavia, per raggiungere questo duplice obiettivo, è anzitutto necessario restituire un "padre" a quel "figlio Telemaco" che, forse, siamo diventati tutti noi: per procedere verso l'*orizzonte dell'avvenire* abbiamo ancora bisogno dei "Maestri testimoni" di cui parlava Paolo VI, di madri che sappiano essere memorie di virtù per i propri figli, di generosi nocchieri che trovino il coraggio di raddrizzare le loro navi, insomma, di *orme nella storia*, giacché, con le parole di un adagio della raccolta, "Senza memoria convocata nel presente non si riesce a procedere verso l'avvenire". E poiché anche la scuola è *fuoco che forgia l'avvenire*, se essa non tradirà il suo fine, che consiste nel formare le persone in crescita affinché diventino *uomini e uomini sapienti e illuminati, aperti alla storia della speranza*, questo "avvenire" potrà veramente essere una realtà più bella; lo garantisce un altro adagio della raccolta: "La realtà più bella è la realizzazione della fantasia degli uomini migliori". Gli uomini migliori sono quelle persone che continuano a guardare in basso mentre salgono, perché sanno che, diversamente, perderebbero di vista la realtà, ma che, pure, invocano *la perseveranza di continuare a cercare il cielo in Terra*; sono gli uomini come Filippo, che quando la barca è piegata da un lato amano sedersi dal lato opposto, e sono le donne come Elvira, che hanno creduto che nel *rinnovellare, riparare e ripensare* consistesse il sale della vita, la regola di un degno stare al mondo. Da persone così viene un messaggio che ci fa bene, un "bene" che ha la stessa sostanza di quello che, come scriveva Dante, "del volere è obietto": "Entra nel nuovo anno di faccia, con lo sguardo tranquillo e il passo calmo, / senza farti scoraggiare dalle sirene dei profeti di sventura. / I problemi si pesano con il bene della vita".

Giovanna Assisi

Gino Scaccia, *Il re della repubblica. Cronaca costituzionale della presidenza di Giorgio Napolitano*, ed. Mucchi, Modena 2015

Quali sono effettivamente i poteri e le prerogative del Presidente della Repubblica Italiana? Perchè la presidenza di Napolitano ha suscitato, accanto ad adesioni e plausi, anche non poche perplessità? Gino Scaccia – autore dei libri: *Il re della repubblica. Cronaca costituzionale della presidenza di Giorgio Napolitano* e *Il Presidente della Repubblica fra evoluzione e trasformazione*, entrambi editi da Mucchi, Modena 2015) – prende in esame gli atti del capo dello Stato, valutandone la conformità alle prescrizioni costituzionali e segnalando i profili di continuità e discontinuità rispetto ai predecessori. Egli prende atto che la Presidenza Napolitano ha rafforzato la tendenza alla crescita di ruolo politico del Quirinale. Si è reso labile il confine tra la direzione politica attiva e la cosiddetta *moral suasion*, ossia i poteri di consiglio e influenza di chi, avendo obblighi di vigilanza e garanzia, induce i soggetti vigilati ad un comportamento moralmente e socialmente corretto, senza ricorrere a quei poteri che la legge mette a sua disposizione per l'esercizio delle funzioni, ma appoggiandosi sull'autorevolezza *super partes* del proprio *status* e sull'importanza del proprio ruolo. L'ampliamento dei poteri del Presidente è stato progressivo, non tanto da un punto di vista formale, in quanto non possiamo dire che siano stati attribuiti poteri nuovi al Capo dello Stato, ma piuttosto da un punto di vista sostanziale. A cambiare sono stati, infatti, l'intensità e la frequenza con cui il Presidente della Repubblica si è avvalso dei suoi poteri per poter mantenere gli equilibri costituzionali. È risultato evidente che la sentenza 1/2013 della Corte Costituzionale, sul conflitto di attribuzione che vedeva contrapposti il Presidente e la Procura di Palermo, in merito alle intercettazioni sui cosiddetti "accordi Stato-mafia", rafforzava il potere del Presidente. Significativo è l'incremento del potere comunicativo, cioè la capacità di far sentire la sua voce in merito alle vicende politiche, economiche e istituzionali, nella forma di potere informativo-conoscitivo (il Presidente deve conoscere a fondo tutte le situazioni politiche e istituzionali per poter intervenire e svolgere la sua funzione di garante costituzionale), il potere di esternazione, tramite manifestazioni pubbliche del suo pensiero, e il potere di persuasione, *moral suasion*, che fa sì che il capo dello Stato, in quanto garante dell'unità nazionale e della Costituzione, intessa rapporti tesi ad armonizzare i conflitti tra i vari organi dello Stato indicando i principi base a cui ispirarsi per risolvere le questioni. In ogni caso, il capo dello Stato è messo in grado di monitorare e controllare tutte le attività e finisce con l'apparire "onnipresente", il che assume per certuni un significato quasi nega-

FIGURA 4: Altre pagine di *Prospettiva Persona*. Notate che le due unghie, posizionate a mano su due *layout page*, sono leggermente disassate.

doci, buttai giù alcune linee guida per ottenere una composizione del genere con LaTeX. Durante il GUIT *meeting 2005*, mentre cercavo qualcuno interessato a collaborare alla realizzazione di un pacchetto del genere, fui presentato a Kaveh Bazargan, al quale descrissi dettagliatamente le linee guida e alcuni algoritmi per ottenere tale composizione. Non sentii più Kaveh, ma qualche tempo dopo vidi grid in cui non ero citato, forse perché avevo descritto tecniche alle quali Kaveh era già arrivato indipendentemente. Comunque, la "mia" ricetta per ottenere una composizione a griglia era, ed è, di eliminare tutta la colla, mantenere un `\baselineskip` costante (certo, per i corpi più piccoli o più grandi questo non è opportuno, ma per `\normalsize`, `\small` e `\large` si può sorvolare sulla minore eleganza compositiva e godere del risultato) e fare in modo che le figure, le tabelle e le formule occupino uno spazio verticale esattamente multiplo di `\baselineskip`, a costo di aggiungere del bianco. Il pacchetto grid risolve brillantemente molte di queste cose, ma non sembra essersi occupato delle minipage che invece *Prospettiva Persona* usa. La soluzione implica inserire la `\minipage` in un box di cui misurare l'altezza e la profondità e poi aggungere uno spazio verticale tale da raggiungere il più vicino multiplo di `\baselineskip`. Vediamo questo comportamento nel codice seguente (alcune linee di codice sono riportate su più righe per motivi di impaginazione):

```
\newenvironment{varreview}[2]{%
% Argomenti
% 1: Citazione di una entry BibLaTeX.
%    Mettere nell'argomento la sola
%    etichetta
% 2: Recensore
\def\recensore{#2}
%Salva il contenuto della minipage
\savebox{\titolorecensione}%
  {\begin{minipage}{\columnwidth}
\vspace{-1.5pt}\begin{tikzpicture}
%Stabilisce la posizione della
%decorazione e le ancora il testo
%come etichetta così che questo
%stia in griglia
\node[anchor=north west] (0cm,0pt)%
  {\begin{minipage}{.97\columnwidth}%
   \hangindent=3mm \fullcite{#1}%
   \phantom{j}\end{minipage}};
%Disegna la decorazione orizzontale
\def\horzpts {90, 87, ..., 21}
\foreach \i in \horzpts%
  \fill [black!\i] (2*(90-\i pt,0)%
     circle (1pt);
%Disegna la decorazione verticale
\def\vertpts {90, 66, 42, 18}
\foreach \i in \vertpts%
  \fill [black!\i] (0, -30+\i/3 pt)%
     circle (1pt);
\end{tikzpicture}
\end{minipage}}
%Pone la minipage nel documento
\noindent\usebox{\titolorecensione}
%Ne valuta altezza e profondità
\settoheight\trvs%
  {\usebox{\titolorecensione}}
\settodepth\trtmp%
  {\usebox{\titolorecensione}}
\advance\trvs by \trtmp
%Toglie lo scartamento della decorazione
\advance\trvs by -4pt
\trvstmp=0pt
%Calcola il minimo \baselineskip
%superiore alla dimensione verticale
%della minipage
\ifdim\trvs=20.5pt\vspace{9pt}\else{
\whiledo{\trvs>\trvstmp}%
  {\advance\trvstmp by \baselineskip}
\advance\trvstmp by -\baselineskip
\advance\trvs by -\trvstmp
%Aggiunge lo scostamento verticale
%più un fattore di aggiustamento
%per compensare la profondità residua
\vspace{-\trvs}
\vspace{.46\baselineskip}}\fi\par
\parindent=3mm
\indent
}
```

Tale soluzione è stata necessaria per le recensioni della rivista, di cui la figura 5 mostra un particolare.

2.2 Indice e gerenza

L'interno di *Prospettiva Persona* si apre con la prima pagina dell'indice. Quest'ultima ha una testatina più alta di quelle che troveremo nel resto della rivista e mostra qualche informazione aggiuntiva al nome della sezione (Indice, appunto): anno di vita della rivista, mese e anno di pubblicazione, numero della rivista nell'anno solare, logo della testata. Tutte le informazioni (escluso il logo della testata) dovranno essere memorizzate in alcune variabili create *ad hoc* sul modello visto in Pignalberi (2015).

L'organizzazione dell'indice è abbastanza razionale: su quattro colonne di ampiezza diversa vengono riportate, da sinista a destra, il nome della rubrica o della rivista interna (solo per il primo articolo di ogni rubrica o nuova rivista), il nome dell'autore o degli autori, il titolo dell'articolo con sotto il titolo inglese e, infine, il numero di pagina. Mentre nella versione InDesign il numero non era preceduto da un *leader*, cioè da una riga di puntini che guida l'occhio tra la fine del titolo e la relativa pagina, nella versione LaTeX abbiamo aggiunto questo elemento.

Per ogni articolo è prevista la presenza (facoltativa) di abstract in italiano e in inglese il cui testo

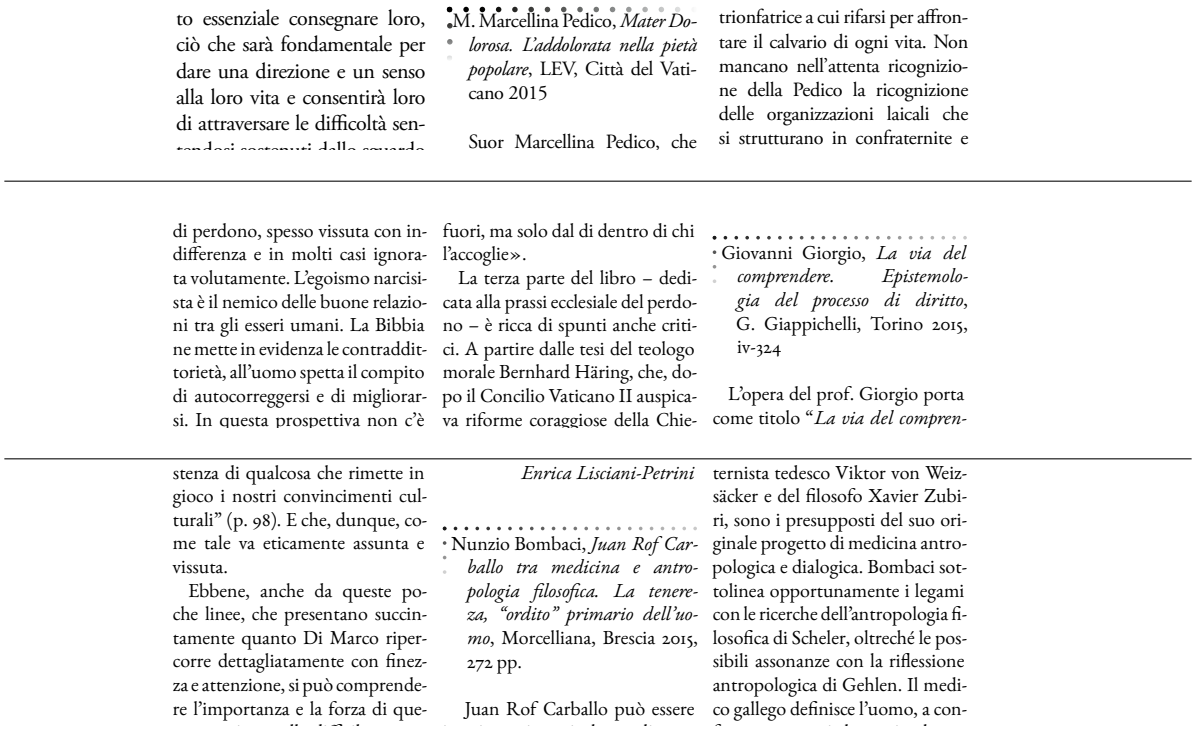


FIGURA 5: Dall'alto in basso: particolare delle recensioni composte con InDesign, con LaTeX senza minipage in griglia e con LaTeX con minipage in griglia. Da notare che, mentre nell'impaginato InDesign il filetto è un pezzo di grafica posticcio con distanza dal testo variabile (in altri punti tale distanza è maggiore), in LaTeX e TikZ il filetto è un nodo e il testo coi dati dell'opera recensita è un'etichetta di tale nodo. Così facendo, TikZ manterrà la distanza grafica-testo sempre costante. Da notare anche che, per imperscrutabili motivi, InDesign non rispetta affatto la griglia. Anche nella prima versione LaTeX la griglia non veniva rispettata; pensavo che bastasse aggiungere alla minipage uno spiazzamento verticale costante, cosa invece errata. Infine non bisogna dimenticare di compensare lo scostamento verticale dovuto alla presenza del filetto e della profondità residua della minipage.

non compare all'inizio dell'articolo ma direttamente nell'indice. L'abstract viene immediatamente dopo le informazioni appena elencate e occupa le due colonne centrali. Uno dei vincoli è fare in modo che solo l'abstract inglese possa andare sulla pagina successiva, con l'abstract italiano che invece resta ancorato agli altri dati. La strategia usata per evitare la separazione proibita è quella di aprire una minipage e porre a *vero* un flag che segnali di aver posto la prima riga di una voce dell'indice. La minipage verrà chiusa quando sarà stato posto il primo abstract[2] e di conseguenza si può porre a *falso* il flag di cui sopra. Anche le singole informazioni della prima riga sono composte entro delle minipage e non dentro una tabella. Pur non rispettando la composizione a griglia, l'indice composto con le minipage ha un aspetto più simile al modello seguito e in generale migliore di quello ottenuto con una tabella. Anche l'inserimento del *leader* è stato agevolato dalle minipage, sebbene il punto di contatto tra la riga che segue il titolo e quella che precede il numero di pagina sia sovente "critico". Mi pare che la soluzione usata con InDesign sia comporre l'indice a mano in una tabella, mentre LaTeX fa tutto da solo.

2. La cosa negativa è che nell'articolo bisogna definire l'abstract, anche se vuoto.

In questo blocco di pagine dev'essere inserito un testo relativo alle norme redazionali. Se c'è abbastanza spazio alla fine dell'indice e questo termina in una pagina destra, detto testo viene inserito di seguito. Se l'indice termina su una pagina sinistra si occupa la destra con un'immagine a pagina piena e le norme vengono sovraimposte all'immagine, curandosi di porre uno sfondo grigio chiaro trasparente tra il testo e l'immagine di sfondo. Se l'indice termina su una pagina destra e non c'è spazio per le norme, le successive due pagine vengono occupate con due metà di un'immagine al vivo all'interno così da risultare una sola immagine larga e le norme vengono poste nella pagina destra con lo stesso sfondo trasparente.

Le successive pagine dedicate alla gerenza e alle affiliazioni sono di semplicissimo ottenimento con tabelle dimensionate *ad hoc* e con multicol.

2.3 Geometrie variabili

La rivista ha due cambiamenti di geometria di pagina. Il primo si ha al termine dell'indice-gerenza (una sorta di *frontmatter*) col passaggio agli articoli: la gabbia del testo subisce un restringimento orizzontale, anche per permettere il posizionamento delle informazioni a margine (da notare che l'ampiezza delle testatine e dei piedini rimane quel-

la del *frontmatter*). Il secondo si ha sempre con il primo articolo (editoriale) di una nuova rivista (quindi seguente a una *splash page*, cfr. sezione 2.4): la prima pagina, e solo quella, di tali articoli è più stretta delle altre ed è a una colonna.

Il cambio di geometria è aiutato dal comando `\newgeometry` del pacchetto geometry. Dovendo stampare la rivista in offset, sono necessarî i crocini di taglio e quindi l'adozione del pacchetto crop.[3] Il cambio di geometria scombina il posizionamento centrale della pagina (nel caso in esame, di dimensione A4) rispetto al foglio (poco meno di un B4) portando tutto in alto a sinistra, a meno di non usare `\CROP@center`. Per far sì che il cambio sia automatico, cioè non debba essere il compositore a inserire i comandi necessarî negli articoli giusti, bisogna fare un po' di controlli sulla pagina di provenienza:

```
\def\@endjournalpart{\vfil\newpage%
  \ifodd\thepage%
  \newgeometry{body={340pt,680pt},%
  left=65pt,right=190pt,headheight=24pt}%
  \else\newgeometry{body={340pt,680pt},%
  left=197pt,right=98pt,headheight=24pt}%
  \fi
  \CROP@center
  \input{\testatalaterale}
}

\newenvironment{papertext}{%
  \if@nj\afterpage%
  {\clearpage\newgeometry%
  {body={436pt,680pt},headheight=24pt}%
  \begin{multicols}{2}%
  \CROP@center}\else%
  \begin{multicols}{2}\fi\global\@njfalse
}
{\end{multicols}\CROP@center\clearpage}
```

Nel primo caso LaTeX ha appena posto una *splash page* (vedi la sezione 2.4), pertanto la pagina successiva avrà una geometria a una colonna stretta. Il controllo sulla (dis)parità della pagina è inutile: ho scoperto dopo aver programmato il comando che la *splash page* va sempre su una pagina destra, quindi la successiva è lapalissianamente sinistra. Questo codice verrà ripulito nelle prossime versioni. Nel secondo caso, invece, usciamo dalla prima pagina dell'editoriale di una nuova rivista e, non sapendo in quale punto ci sarà il cambio, sfruttiamo afterpage per il cambio.

Ho avuto un problema relativo al cambio di geometria proprio negli editoriali delle riviste inserite in *Prospettiva Persona*. Il paragrafo a cavallo della prima e della seconda pagina veniva sì interrotto correttamente, ma la porzione di testo sulla seconda pagina manteneva la stessa larghezza della prima pagina, col risultato di sovrapporsi al testo nella colonna di destra. Per qualche motivo che non ho capito l'uso di `\linebreak` era inutile; ho dovuto quindi creare un comando (`\interrompi`) la cui sintassi è la seguente:

```
\newcommand\interrompi{%
\parfillskip\vadjust{\break}%
\newpage\noindent\CROP@center}
```

Tale comando non è in grado di sillabare l'ultima parola della prima pagina, pertanto l'ultima riga può risultare brutta da vedere. Ora il lettore attento si chiederà: «Non potevi legare il cambio di geometria al comando `\interrompi`?». Avrei potuto, ma 1) `\interrompi` è stato programmato in seguito; 2) se un capoverso termina esattamente alla fine della prima pagina a una colonna non servirà chiamare il comando. Se il cambio di geometria fosse legato all'uso di `\interrompi`, questo andrebbe usato sempre e comunque, peraltro peggiorando l'aspetto dell'ultima riga a causa di `\parfillskip`.

2.4 *Splash page*

Ogni rivista ha una copertina. Non fanno eccezione le riviste interne a *Prospettiva Persona*, che hanno una copertina con tanto di testata e strilli. Detta copertina non è una copertina vera e propria ma solo una prima di copertina stampata sulla stessa carta di tutte le pagine interne della rivista. Per questo è assimilabile a una *splash page*, la pagina iniziale di una storia a fumetti spesso occupata da un'unica vignetta a tutta pagina. Questa copertina ha un'immagine al vivo, che quindi deve uscire in tutte e quattro le abbondanze e, sovraimposte a essa, la testata e gli strilli. Siccome la posizione della testata e degli strilli non è fissa ma deve trovare collocazione nei punti in cui non disturba o non viene disturbata dal contenuto dello sfondo, bisogna fare in modo che sia possibile dare le coordinate per posizionare questi elementi. Una versione più recente della classe permette anche di sovraimporre un ulteriore logo, anch'esso con le proprie coordinate. Tutte queste informazioni – nomi dei file e coordinate – devono essere passate al comando come parametri. Sappiamo che un comando TeX prende fino a nove parametri. A noi ne servono di più (il primo è facoltativo) quindi dobbiamo ricorrere a uno stratagemma:

```
\newcommand\journalpart{%
% Prende in input 13 parametri
...
  \secdef\@journalpart\@sjournalpart}
\newcommand\@journalpart[9][]{%
  \def\shortttl{#1}
  \def\longttl{#2}
  \def\bcgndimg{#3}
  \def\ttlimg{#4}
  \def\xttlimg{#5}
  \def\yttlimg{#6}
```

3. Bisogna notare che anche geometry è in grado di aggiungere i crocini di taglio alle pagine tramite l'opzione showcrop.

Figura 6: Unghia per le pagine dispari di *Prospettiva Persona*.

```
  \def\shttxt{articoli/#7}
  \def\xshttxt{#8}
  \def\yshttxt{#9}
  \@journalpartcntd
}
\newcommand\@journalpartcntd[4]{%
  \def\testatalaterale{Costanti/#1}
  \def\superim{#2}
  \def\xsuperim{#3}
  \def\ysuperim{#4}
...
}
\def\@sjournalpart#1{\@endjournalpart}
```

Nel punto del documento in cui chiamiamo il comando appena visto scriveremo qualcosa del tipo:

```
\journalpart{Prospettiva Donna}%
{immagini/CopPd}%
{PDonnagrandeB.pdf}{20}{20}%
{ProspettivaDonnaStrilli}{50}{680}%
{ProspettivaDonnaGerenza}%
{immagini/Trifogli.JPG}{100}{100}
```

2.5 Decorazioni

Per ottenere i filetti e le unghie ho trovato comodissimo Ti*k*Z, che permette di disegnare algoritmicamente con un codice molto compatto. Per ottenere l'unghia rappresentata nella figura 6 basta usare il codice seguente:

```
\begin{tikzpicture}
\node (0,0) {
\begin{minipage}{1cm}
\fontsize{26}{12}\selectfont
\textit{P\hspace{-1.17ex}%
  \raisebox{-12pt}{P}}
\end{minipage}};
\shade[right color=lightgray,%
  left color=white] (.5,.525)%
  rectangle +(1.2,-1.015);
\def\horzpts{-4pt, 0pt, 4pt,%
  8pt, 12pt, 16pt, 20pt, 24pt,%
  28pt, 32pt, 36pt, 40pt, 44pt, 48pt}
\foreach \x in \horzpts%
 \fill [black!96] (\x,-.6) circle (1pt);
\foreach \i in {96, 92, ..., 12}%
  {\fill [black!\i]%
   ($(-4pt,0)!.6!3.75*\i:(-4pt,-1)$)%
   circle (1pt);}
\end{tikzpicture}
```

Non altrettanto comodo l'hanno trovato i tipografi che lamentano il fatto che il grigio dei filetti e delle unghie non è un grigio reale ma, stando a quanto riscontrato da Illustrator, è la combinazione CMYK di ciano, magenta, giallo e nero. Inkscape, invece, mi dice essere la risultanza della combinazione RGB di rosso, verde e blue in percentuali uguali.[4] Vogliono solo le immagini a colori, il resto dev'essere *grayscale*. Finora non ho trovato una soluzione; neanche l'opzione gray di xcolor risolve. Prima di scrivere all'autore di pgf, mi rimane solo da testare l'ultima versione del pacchetto presa direttamente su SourceForge. La soluzione adottata per i due numeri già pubblicati è stata duplice: quando possibile, abbiamo generato dei pdf coi soli filetti o unghie ridotti da colore a grigio tramite PitStop e quindi inclusi come immagine anziché generati a ogni compilazione; nel caso delle recensioni, per evitare tutti i problemi di posizionamento e calcolo degli ingombri verticali, sono state trattate con PitStop tutte le relative pagine.

Mentre i filetti vengono posizionati in punti fissi della rivista (nelle testatine e al lato delle gerenze degli editoriali), le unghie godono di un trattamento speciale: la loro posizione orizzontale è fissa ed è determinata dal limite della pagina e dalla necessità che parte del disegno finisca nell'abbondanza cosicché, quando quest'ultima viene tagliata, il grigio a bordo pagina risulta evidente; la posizione verticale aumenta verso il basso al cambiare delle rubriche o delle riviste interne. Dunque i comandi `\part` e `\journalpart` devono fare in modo che venga gestita la posizione verticale delle unghie oltre al materiale di propria competenza:

```
\newcounter{pospart}
\setcounter{pospart}{150}
...
\newcommand\part{%
  \clearpage
  \addtocounter{pospart}{35}
...
}
```

mentre i comandi `\sidetitle` e `\sidetitleonly` si occupano del posizionamento:

```
\newcommand\sidetitle{%
...
  \begin{textblock}{100}(-13,\thepospart)
    \includegraphics{unghiapari.pdf}
  \end{textblock}
}
```

Le pagine pari della rivista hanno un'ulteriore "decorazione": una specie di nota a margine che riporta la testata in piccolo, il numero della rivista, l'anno solare, il numero della rivista nell'anno solare e le pagine iniziale e finale dell'articolo corrente. Mentre delle prime informazioni (anno, mese, numero di rivista) abbiamo già detto e sono statiche,

4. In effetti, Till Tantau mi ha confermato che Ti*k*Z trasforma anche i grigi in RGB indipendentemente dallo spazio di colore richiesto.

valgono cioè per tutto il numero, le ultime due sono dinamiche e variano articolo per articolo. Vediamo il codice che consente la loro determinazione:

```
\def\arbitrarypagelabel#1#2{\@bsphack
  \protected@write\@auxout{}%
         {\string\newlabel{#1}%
         {{\@currentlabel}{#2}}}%
  \@esphack}
\newcounter{lptmp}
\def\@makepaperhead#1#2#3{%
\renewcommand\and{\textbullet~}
\addtocounter{paperno}{1}
\setcounter{lptmp}{\thepage}
\label{articolofp\thepaperno}
\arbitrarypagelabel%
{articololp\thepaperno}%
  {\pageref{LastPage}}
\ifthenelse{\thepaperno>1}%
  {\addtocounter{paperno}{-1}%
   \addtocounter{lptmp}{-1}%
   \arbitrarypagelabel%
{articololp\thepaperno}%
   {\thelptmp}\addtocounter{paperno}{1}}%
   {}
...
}
```

Per ogni nuovo articolo viene aumentato di 1 il contatore degli articoli e si pone il contatore temporaneo di pagina al numero corrente di pagina. Si crea un'etichetta col nome `articolofp` seguito dal numero di articolo (simuliamo un array) che referenzia[5] quindi la prima pagina dell'articolo. Successivamente creiamo un'altra etichetta (`articololp` con lo stesso criterio ma con contenuto arbitrario perché inizialmente non sappiamo quando l'articolo terminerà) che referenzierà il contenuto di `LastPage`, contenuto che aumenta man mano che la compilazione procede. A meno che un articolo non sia il primo, sappiamo che l'ultima pagina dell'articolo che lo precede è pari alla prima dell'articolo corrente -1. È quanto fatto nell'ultimo `\ifthenelse`. Il primo e l'ultimo articolo hanno rispettivamente la prima e l'ultima pagina perfettamente etichettate. Ora che abbiamo tutti i dati, è immediato richiamarli man mano sempre in `\sidetitle` e `\sidetitleonly`:

```
\newcommand\sidetitle{%
  \begin{textblock}{62}(30,130)
    \begin{minipage}{60pt}
      \centering
      \textsc{\small ProspettivA}%
      \\[-.5\baselineskip]
      \textsc{\raisebox{-2pt}%
      {\textbullet}\,%
      {\footnotesize persona}\,%
      \raisebox{-2pt}{\textbullet}}%
      \textit{\scriptsize%
      \liningnums{\@nvolume} %
      (\liningnums{\@journalyear}/%
      \liningnums{\@quarter}), %
      \liningnums{\ifthenelse%
      {\pageref{articolofp\thepaperno}=%
       \pageref{articololp\thepaperno}}%
      {\pageref{articolofp\thepaperno}}%
      {\pageref{articolofp\thepaperno}-%
      \pageref{articololp\thepaperno}}}}
    \end{minipage}
  \end{textblock}
...
}
```

Questa soluzione è stata implementata tra le prime caratteristiche della classe. Siccome in seguito ho deciso di delimitare il corpo di un articolo con un ambiente denominato papertext, una soluzione più semplice sarà porre l'etichetta di fine articolo al momento della chiusura dell'ambiente.

2.6 Bibliografia

Una delle ragioni principali per cui il redattore di *Prospettiva Persona* ha voluto abbandonare la composizione con InDesign in favore di LaTeX è stata per la coerenza intrinseca delle bibliografie. Per motivi di stile, la scelta degli strumenti è stata biber, biblatex e philosophy-verbose. Durante i primi test abbiamo riscontrato un'incompatibilità di biblatex-philosophy con fancyhdr. Ivan Valbusa, l'autore dell'estensione di biblatex, ha prontamente corretto quell'incompatibilità.

3 Risultato

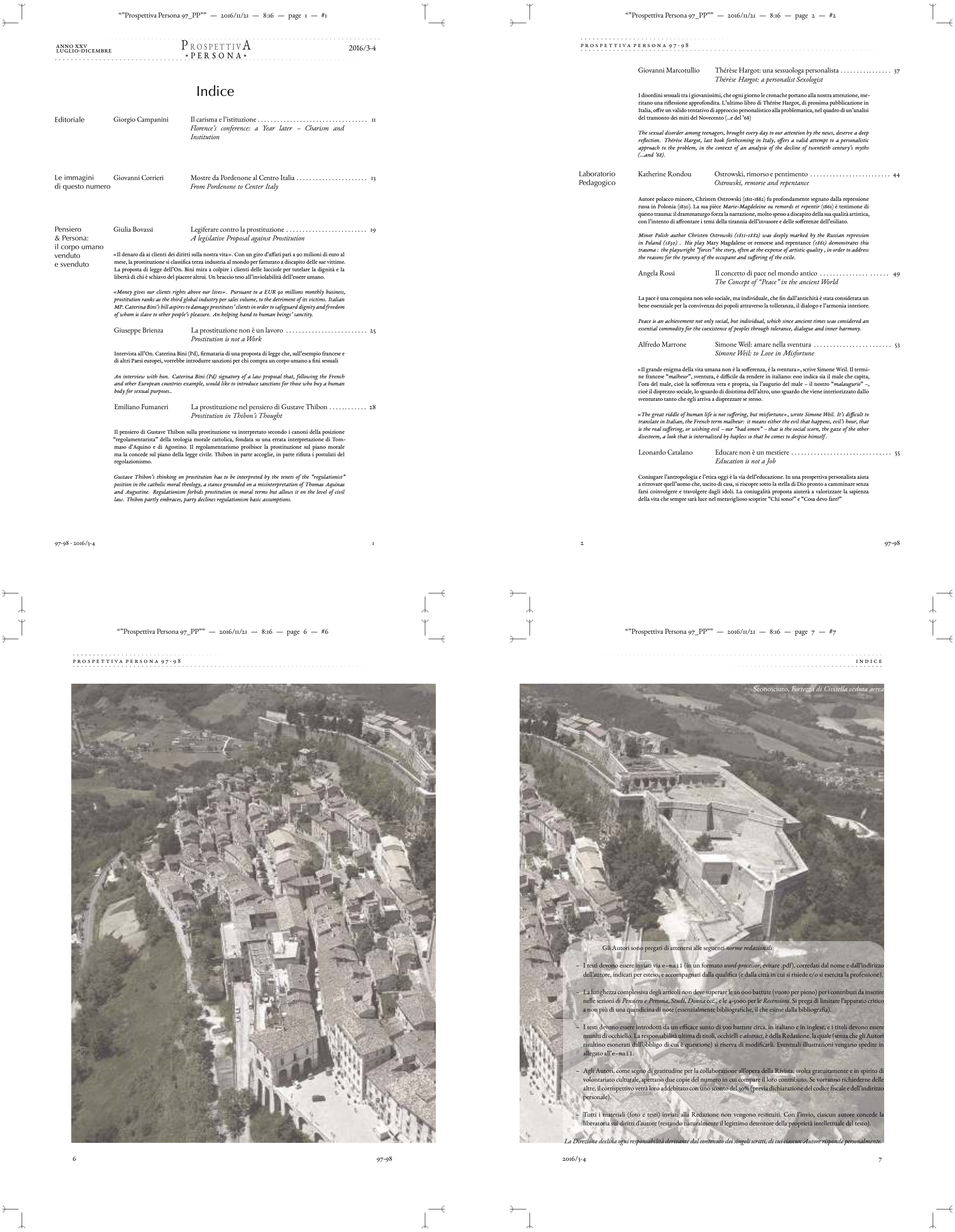
Il primo numero prodotto era ancora affllitto da alcuni bug. Il primo a essere riscontrato stava proprio a pagina 1: la testatina non veniva stampata perché non avevo eliminato tutti i `\pagestyle{empty}` che confliggevano con lo style `plain` impostato per la prima pagina. Un altro bug fastidiosissimo era l'errata disposizione della gabbia di pagina e dei piedini della prima pagina di ogni articolo. Colpa doppiamente mia perché avevo definito l'altezza delle testatine troppo bassa e perché non avevo notato il messaggio di *warning* del compilatore a tale proposito. Ci sono stati anche altri bug, di cui parlerò diffusamente nell'articolo lungo.

Potete vedere alcune pagine del "secondo" numero nelle figure 7–10.

Conclusioni

LaTeX è in grado di rivaleggiare con successo con InDesign anche in quei campi che sono apparentemente fuori dalla sua portata. Pagine con colonne multiple composte a griglia ed elementi decorativi

5. Claudio Beccari mi ha fatto notare la possibile erroneità e l'essere un *false friend* di questo termine. Sono d'accordo con lui, ma è ormai il termine d'elezione quando si parla di puntatori. Quindi lo uso dal momento che il meccanismo `\label`-`\ref` è assimilabile ai puntatori.



“”Prospettiva Persona 97_PP”” — 2016/11/21 — 8:16 — page 1 — #1

ANNO XXV
LUGLIO-DICEMBRE

PROSPETTIVA
• PERSONA •

2016/3-4

Indice

Editoriale — Giorgio Campanini — Il carisma e l’istituzione 11
Florence’s conference: a Year later – Charism and Institution

Le immagini di questo numero — Giovanni Corrieri — Mostre da Pordenone al Centro Italia 13
From Pordenone to Center Italy

Pensiero & Persona: il corpo umano venduto e svenduto — Giulia Bovassi — Legiferare contro la prostituzione 19
A legislative Proposal against Prostitution

«Il denaro dà ai clienti dei diritti sulla nostra vita». Con un giro d’affari pari a 90 milioni di euro al mese, la prostituzione si classifica terza industria al mondo per fatturato a discapito delle sue vittime. La proposta di legge dell’On. Bini mira a colpire i clienti delle lucciole per tutelare la dignità e la libertà di chi è schiavo del piacere altrui. Un braccio teso all’inviolabilità dell’essere umano.

«Money gives our clients rights above our lives». Pursuant to a EUR 90 millions monthly business, prostitution ranks as the third global industry per sales volume, to the detriment of its victims. Italian MP. Caterina Bini’s bill aspires to damage prostitutes’ clients in order to safeguard dignity and freedom of whom is slave to other people’s pleasure. An helping hand to human beings’ sanctity.

Giuseppe Brienza — La prostituzione non è un lavoro 25
Prostitution is not a Work

Intervista all’On. Caterina Bini (Pd), firmataria di una proposta di legge che, sull’esempio francese e di altri Paesi europei, vorrebbe introdurre sanzioni per chi compra un corpo umano a fini sessuali

An interview with hon. Caterina Bini (Pd) signatory of a law proposal that, following the French and other European countries example, would like to introduce sanctions for those who buy a human body for sexual purposes..

Emiliano Fumaneri — La prostituzione nel pensiero di Gustave Thibon 28
Prostitution in Thibon’s Thought

Il pensiero di Gustave Thibon sulla prostituzione va interpretato secondo i canoni della posizione “regolamentarista” della teologia morale cattolica, fondata su una errata interpretazione di Tommaso d’Aquino e di Agostino. Il regolamentarismo proibisce la prostituzione sul piano morale ma la concede sul piano della legge civile. Thibon in parte accoglie, in parte rifiuta i postulati del regolazionismo.

Gustave Thibon’s thinking on prostitution has to be interpreted by the tenets of the “regulationist” position in the catholic moral theology, a stance grounded on a misinterpretation of Thomas Aquinas and Augustine. Regulationism forbids prostitution in moral terms but allows it on the level of civil law. Thibon partly embraces, party declines regulationism basic assumptions.

97-98 - 2016/3-4 1

“”Prospettiva Persona 97_PP”” — 2016/11/21 — 8:16 — page 2 — #2

PROSPETTIVA PERSONA 97-98

Giovanni Marcotullio — Thérèse Hargot: una sessuologa personalista 37
Thérèse Hargot: a personalist Sexologist

I disordini sessuali tra i giovanissimi, che ogni giorno le cronache portano alla nostra attenzione, meritano una riflessione approfondita. L’ultimo libro di Thérèse Hargot, di prossima pubblicazione in Italia, offre un valido tentativo di approccio personalistico alla problematica, nel quadro di un’analisi del tramonto dei miti del Novecento (...e del ’68)

The sexual disorder among teenagers, brought every day to our attention by the news, deserve a deep reflection. Thérèse Hargot, last book forthcoming in Italy, offers a valid attempt to a personalistic approach to the problem, in the context of an analysis of the decline of twentieth century’s myths (...and ’68).

Laboratorio Pedagogico — Katherine Rondou — Ostrowski, rimorso e pentimento 44
Ostrowski, remorse and repentance

Autore polacco minore, Christen Ostrowski (1811-1882) fu profondamente segnato dalla repressione russa in Polonia (1830). La sua pièce *Marie-Magdeleine ou remords et repentir* (1861) è testimone di questo trauma: il drammaturgo forza la narrazione, molto spesso a discapito della sua qualità artistica, con l’intento di affrontare i temi della tirannia dell’invasore e delle sofferenze dell’esiliato.

Minor Polish author Christen Ostrowski (1811-1882) was deeply marked by the Russian repression in Poland (1830) . His play Mary Magdalene or remorse and repentance *(1861) demonstrates this trauma : the playwright ”forces” the story, often at the expense of artistic quality , in order to address the reasons for the tyranny of the occupant and suffering of the exile.*

Angela Rossi — Il concetto di pace nel mondo antico 49
The Concept of “Peace” in the ancient World

La pace è una conquista non solo sociale, ma individuale, che fin dall’antichità è stata considerata un bene essenziale per la convivenza dei popoli attraverso la tolleranza, il dialogo e l’armonia interiore.

Peace is an achievement not only social, but individual, which since ancient times was considered an essential commodity for the coexistence of peoples through tolerance, dialogue and inner harmony.

Alfredo Marrone — Simone Weil: amare nella sventura 53
Simone Weil: to Love in Misfortune

«Il grande enigma della vita umana non è la sofferenza, è la sventura», scrive Simone Weil. Il termine francese “*malheur*”, sventura, è difficile da rendere in italiano: esso indica sia il male che capita, l’ora del male, cioè la sofferenza vera e propria, sia l’augurio del male – il nostro “*malaugurio*” –, cioè il disprezzo sociale, lo sguardo di disistima dell’altro, uno sguardo che viene interiorizzato dallo sventurato tanto che egli arriva a disprezzare se stesso.

«The great riddle of human life is not suffering, but misfortune», wrote Simone Weil. It’s difficult to translate in Italian, the French term malheur: it means either the evil that happens, evil’s hour, that is the real suffering, or wishing evil – our “bad omen” – that is the social scorn, the gaze of the other disesteem, a look that is internalized by hapless so that he comes to despise himself.

Leonardo Catalano — Educare non è un mestiere 55
Education is not a Job

Coniugare l’antropologia e l’etica oggi è la via dell’educazione. In una prospettiva personalista aiuta a ritrovare quell’uomo che, uscito di casa, si riscopre sotto la stella di Dio pronto a camminare senza farsi coinvolgere e travolgere dagli idoli. La coniugalità proposta aiuterà a valorizzare la sapienza della vita che sempre sarà luce nel meraviglioso scoprire “Chi sono?” e “Cosa devo fare?”

2 97-98

“”Prospettiva Persona 97_PP”” — 2016/11/21 — 8:16 — page 6 — #6

PROSPETTIVA PERSONA 97-98

6 97-98

“”Prospettiva Persona 97_PP”” — 2016/11/21 — 8:16 — page 7 — #7

INDICE

Sconosciuto, Fortezza di Civitella veduta aerea

Gli Autori sono pregati di attenersi alle seguenti *norme redazionali*:

- I testi devono essere inviati via `e-mail` (in un formato *word-processor*, evitare .pdf), corredati dal nome e dall’indirizzo dell’autore, indicati per esteso, e accompagnati dalla qualifica (e dalla città in cui si risiede e/o si esercita la professione).
- La lunghezza complessiva degli articoli non deve superare le 20 000 battute (vuoto per pieno) per i contributi da inserire nelle sezioni di *Pensiero e Persona*, *Studî*, *Donna* ecc., e le 4-5000 per le *Recensioni*. Si prega di limitare l’apparato critico a non più di una quindicina di note (essenzialmente bibliografiche, il che esime dalla bibliografia).
- I testi devono essere introdotti da un efficace sunto di 500 battute circa, in italiano e in inglese, e i titoli devono essere muniti di occhiello. La responsabilità ultima di titoli, occhielli e *abstract*, è della Redazione, la quale (senza che gli Autori risultino esonerati dall’obbligo di cui è questione) si riserva di modificarli. Eventuali illustrazioni vengano spedite in allegato all’`e-mail`.
- Agli Autori, come segno di gratitudine per la collaborazione all’opera della Rivista, svolta gratuitamente e in spirito di volontariato culturale, spettano due copie del numero in cui compare il loro contributo. Se vorranno richiederne delle altre, il corrispettivo verrà loro addebitato con uno sconto del 30% (previa dichiarazione del codice fiscale e dell’indirizzo personale).
- Tutti i materiali (foto e testi) inviati alla Redazione non vengono restituiti. Con l’invio, ciascun autore concede la liberatoria sui diritti d’autore (restando naturalmente il legittimo detentore della proprietà intellettuale del testo).

La Direzione declina ogni responsabilità derivante dal contenuto dei singoli scritti, di cui ciascun Autore risponde personalmente.

2016/3-4 7

FIGURA 7: Alcune pagine di *Prospettiva Persona* prodotte con LaTeX.

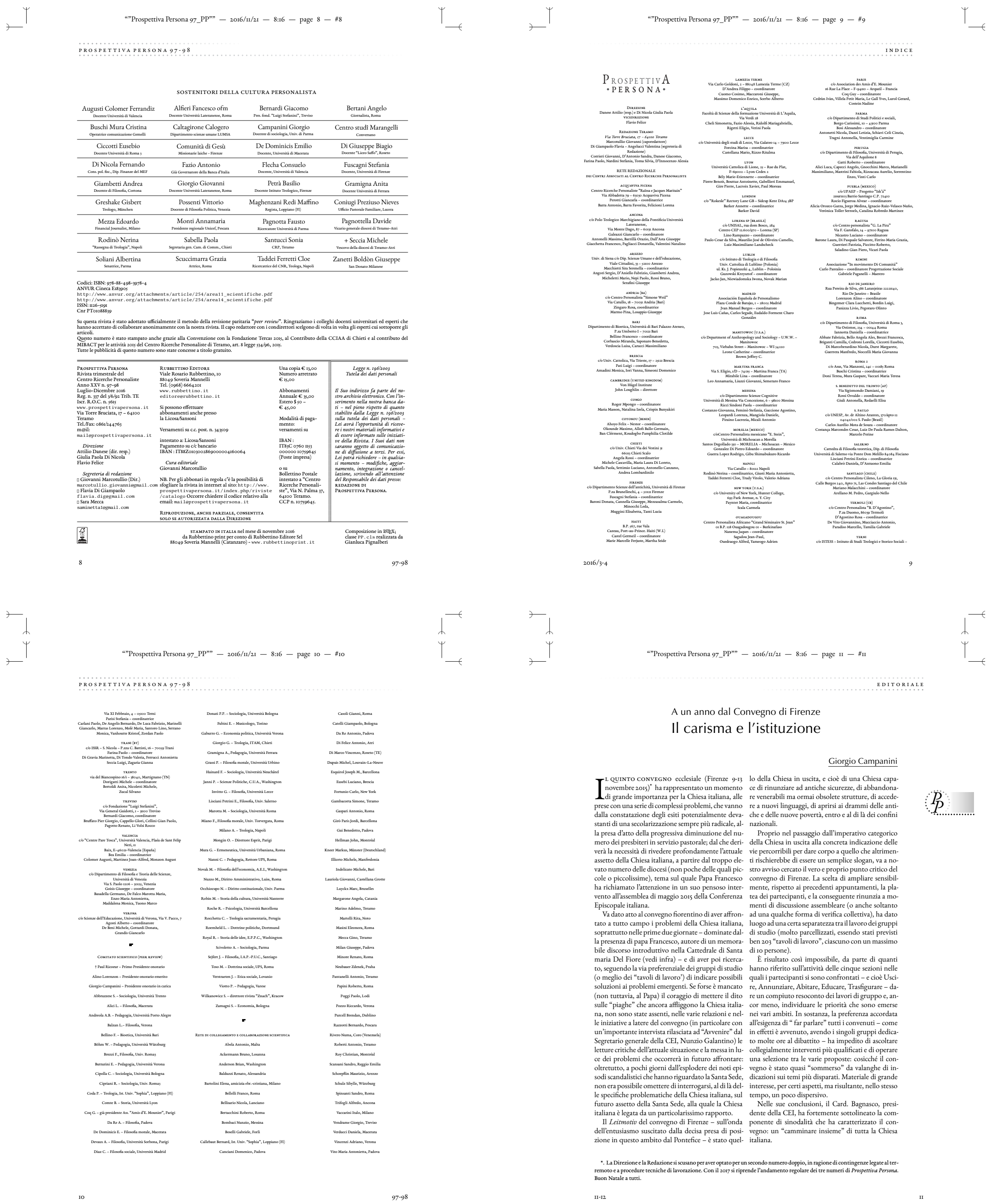


FIGURA 8: Altre pagine di *Prospettiva Persona* prodotte con LATEX.

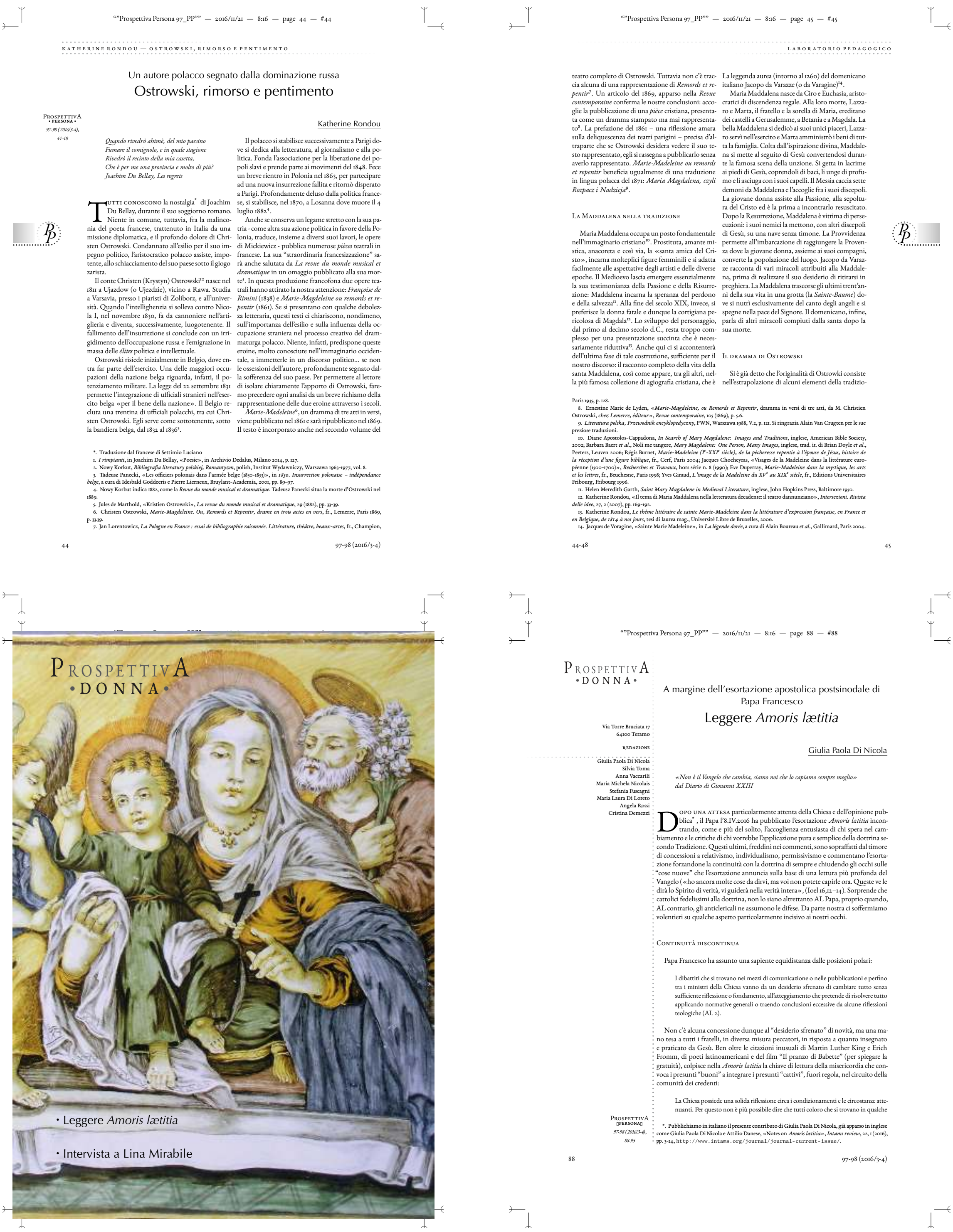
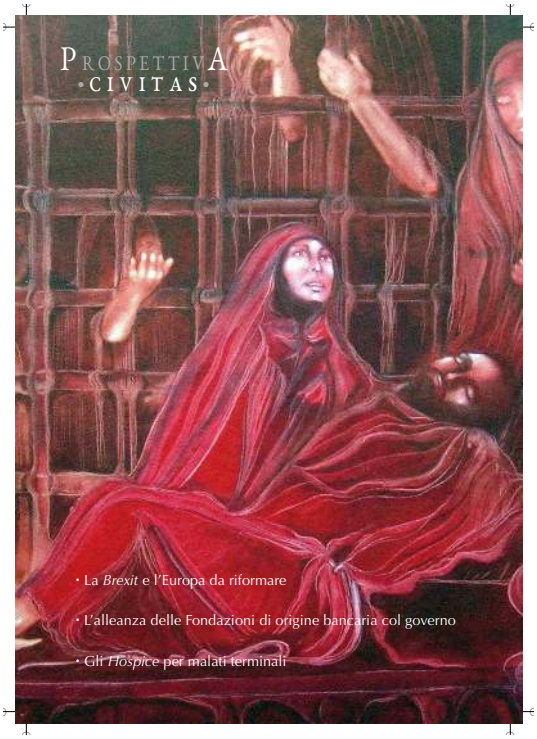


FIGURA 9: Altre pagine di *Prospettiva Persona* prodotte con LaTeX.



Recensioni

• Roberto Gabellini, *L'ultima marcia del tenente Péguy*, 1ª ed., Ares, Milano 2014

Scrivere un poema su di un poeta, ritmarlo, con una sequenza di testi di varia lunghezza, composti in quartine proprio come Eva di Péguy, sugli avvenimenti storici e biografici di poco più di un mese degli ultimi giorni del poeta, rigorosamente accertati ed insieme cogliere il significato profondo di quella morte in battaglia, è stato il modo migliore per Roberto Gabellini per ricordare il centenario del sacrificio di Charles Péguy il 5 settembre 1914. Il poema è preceduto da una introduzione di Alessandro Rivali e chiuso da un saggio di Pigi Colognesi, Una breve biografia a cura di Flora Crescini e una dettagliata cronologia dei fatti accaduti dal 1 agosto 1914 al 5 settembre 1914 permettono di comprendere ogni momento di questa narrazione poetica, che trova la sua chiave interpretativa nelle parole che lo stesso Péguy, poco prima di imbarcarsi sul treno che lo portava a fronte, disse a Geneviève Favre, la madre di Jacques Maritain: "Parto soldato della Repubblica, per il disarmo generale e l'ultima guerra". Questo testo è inserito nel poema di Gabellini, come altri frammenti delle poesie di Péguy, che vengono amalgamati nella interpretazione degli avvenimenti di cui si parla in un dialogo continuo ed affettuoso con il tenente Péguy.

Péguy socialista non rinuncia al suo ideale di pace, rifiuta il pacifismo utopistico, sa che bisogna difendere la Repubblica e spera che la tragedia imminente sia l'ultima guerra, convinto che una democrazia reale possa migliorare le condizioni di vita del popolo. Scrive Bolognesi «per lui la Repubblica significava la sintesi degli ideali del popolo francese, quelli della rivoluzione, della democrazia, del socialismo; in definitiva quelli della "Città armoniosa" da lui sempre perseguita» (p. 140). Ma con il dramma collettivo dei popoli in guerra di annoda con una crisi personale, che Gabellini analizza con tenerezza nei suoi versi. Péguy milita nel Partito socialista e dirige una rivista socialista. Nel 1897 sposa con matrimonio civile Charlotte Badouin, sorella del suo migliore amico, da cui avrà diversi figli. Scrive articoli, molto violenti per difendere l'ufficiale ebreo Dreyfus ingiustamente condannato. Ma quando il Partito decide una censura preventiva sugli articoli, lascia il partito e fonda i "*Cahiers de la Quinzaine*", una rivista indipendente che dirige tra mille difficoltà finanziarie. Nel 1908 ritorna alla fede cattolica, si confida con Maritain ed un vecchio amico, diventato benedettino, Louis Baillet che lo informano, della necessità per essere cattolico di regolarizzare con il sacramento il suo matrimonio e di battezzare i figli. La moglie non è d'accordo e Péguy per rispettare la sua libertà rinuncia a rimane fedele al suo matrimonio, anche quando si innamora di Blanche una giovane ebrea che lo aiuta ai "Cahiers" e ispira la sua poesia; la rispetta e rimane fedele al suo matrimonio. Commenta Bolognesi: «In un paradosso che solo la concezione cristiana di verginità riesce a contemplare, Péguy fu contemporaneamente fedele alla moglie e alla donna dei suoi sogni (che, tra l'altro aiutò a trovare un buon marito). Fu probabilmente per tale purezza, unita ad un profondo rispetto, che Charlotte, dopo la morte del marito si convertirà al cristianesimo e farà battezzare i figli» (pp. 137-138).

Quest'avventura politica, questa travaglio affettivo che non ha nulla delle svenevolezze romantiche, questa fede, senza sacramenti, sofferta nel mistero della grazia di Dio è tradotto da Gabellini in testi poetici scanditi in quindici giornate di concreti avvenimenti spirituali. Accenno a qualche passaggio particolarmente significativo.

1 agosto 1914. La Francia, a causa della dichiarazione di guerra della Germania, decreta la mobilitazione generale, inizia il primo conflitto mondiale

> Come ci si prepara per partire,
> quali sono
> le parole con cui poterci salutare,
> dire addio, e offrire una speranza
> senza dire una bugia; come si fa
> a mettere in ordine la vita,
> a non lasciare nulla di sospeso, il lavoro,
> il futuro dei tuoi figli, il loro pane quotidiano;
> a lasciare i vecchi, soli (pag. 17).

Péguy deve lasciare la moglie Charlotte, che attende il quarto figlio, è preoccupato ed inquieto, li ha affidati all'attenzione del suo maestro Henri Bergson. I versi struggenti di Gabellini fanno comprendere che forse Charlotte sarebbe disposta a fare battezzare il figlio nascituro, il tenente Péguy chiede solo un pellegrinaggio a Chartres e paragona il camminare dei pellegrini alla marcia dei soldati che stanno accorrendo al fronte per difendere la Repubblica:

> In questi momenti si potrebbe

PROSPETTIVA •CIVITAS•

Via Sistina 121
00187 Roma
civitas@prospettivapersona.it

REDAZIONE
Flavio Felice (coordinatore)
Paolo Asolan
Fabio G. Angelini
Mauro Bontempi
Antonio Campati
† Sergio Lanza
Anna Maria Merlini
Maurizio Serio
Fiore Zuccarini

L'inserto "Prospettiva Civitas" è stato realizzato con il contributo del Centro Studi Tocqueville-Acton e grazie alla Convenzione con la Fondazione Terras 2014

PROSPETTIVA (PERSONA)
97-98 (2016/3-4), 101-103

L'oblio delle nostre radici liberali e cristiane

La *Brexit* e l'Europa da riformare

Flavio Felice

Premesso che, Brexit o non Brexit, un europeista come il sottoscritto sarà sempre europeista, non possiamo non registrare una forte battuta d'arresto del progetto d'integrazione europea e la profonda crisi nella quale ormai esso sembra essere caduto. Di fronte ad un certo entusiasmo manifestato dalla comunità euroscettica mondiale, esaltata e confermata nel proprio scetticismo dalla decisione dei sudditi di "Sua Maestà" di abbandonare l'UE, ho provato un magone allo stomaco; la sintetica riflessione che segue è il frutto delle discussioni avute in questi giorni con gli amici del Centro Studi Tocqueville-Acton, con gli studenti e i docenti del "*Free Society Seminar*" di Bratislava e con i colleghi Dario Antiseri, Rocco Buttiglione, Dario Velo e Michael Wohlgemuth.

Invero, non si può dimenticare come, nelle prove della storia, il Regno Unito si sia dimostrato la patria del liberalismo e la culla dell'istituzione parlamentare, il bastione della libertà europea, mentre il Continente si attardava in autoritarismi, totalitarismi e guerre nazionaliste e fratricide. È per questo che oggi l'UE è indubbiamente più povera, almeno in termini di cultura politica liberale, e meno attrezzata per affrontare con coraggio le sfide della competizione globale.

D'altro canto, occorre anche chiedersi quanto questa Gran Bretagna conservi di quella cultura e quanto invece non l'abbia rinnegata, assumendo una posa politica di negazione pragmaticamente ottusa delle quattro libertà fondamentali dell'Europa unita: movimento di merci, servizi, capitali e persone. Negando specie quest'ultima, si cade nel più banale protezionismo, arroccandosi intorno ad una anacronistica nozione sovranità nazionale.

Eppure non mancavano anticorpi a questa deriva: tra i tanti, voglio menzionare la lezione dell'economista federalista e liberale Lionel Robbins e del suo collega austriaco Friedrich August von Hayek. Entrambi hanno sognato un contributo all'edificazione dell'Europa completamente diverso dagli esiti odierni. Nei loro saggi, rispettivamente, L'anarchia internazionale e l'economia liberale e L'anarchia internazionale e l'economia socialista del 1937 e Le condizioni economiche del federalismo tra gli Stati (1939), ora in uscita per i tipi Rubbettino, Robbins e Hayek mostrano l'impossibilità, per un sistema liberale, di sopravvivere all'interno di un sistema internazionale fondato sulla rigida divisione in Stati nazionali, evidenziando come la pianificazione della produzione a livello internazionale, e la progressiva perdita della libertà economica, sviluppi una corrente mirante a sostituire le forme democratiche di un'organizzazione con un sistema dittatoriale; di qui l'esigenza, avvertita dai fautori del "liberalismo ordinamentale" e dell'economia sociale di mercato, di una costituzione economica che promuova la concorrenza su scala globale.

Per entrambi, l'obiettivo è l'edificazione di un sistema mondiale capace di adattarsi ai mutamenti e tale da fornire incentivi all'adattamento, attraverso un complesso di istituzioni idonee ad affrontare le difficoltà dell'organizzazione su scala mondiale. Questa insistenza sulle istituzioni è ciò che distingue il loro liberalismo da quella caricatura che lo intende come mero sistema anarchico, privo di regole, nel quale regna il disordine che avvantaggia gli individui senza alcuno scrupolo.

Ma non sono soli: con loro, anche Wilhelm Röpke, Luigi Einaudi e Luigi Sturzo, si schierarono apertamente a favore di un'organizzazione sopranazionale sul modello

PROSPETTIVA •PERSONA•
97-98 (2016/3-4), 219-237

> ottenere
> qualsiasi cosa, vincolare ogni cuore,
> legare gli amici, la sposa, anche i figli
> a qualsiasi promessa.
> Te l'ha offerto lei stessa, ti ama;
> avresti almeno un figlio battezzato, e tu,
> così, cristiano vero, finalmente accettato
> anche se morto; ma non sarebbe giusto
> non per te almeno. Le hai chiesto solo una preghiera,
> una, alla tua maniera: una ogni anno,
> a Chartres, camminando;
> quasi marciando (p. 21).

Ed in questa stessa pagina c'è anche lo snodo interiore di questa avventura spirituale, bisogna conservare la fede e lasciare lavorare la grazia, che i come l'acqua continua a colare, a gocciare, a scava la sua strada. Bisogna accettare la separazione dai propri cari e la solitudine. La salvezza non è opera nostra ma di Dio.

> La tristezza che ti opprime; tua moglie e i tuoi figli,
> loro sì, insieme, come un'altra famiglia,
> non più la tua, che ormai vive da sola;
> lascia la tua fede, qui, da qualche parte,
> magari nascosta in un cappotto ad aspettare,
> senza farsi vedere, il giorno in cui, chissà,
> qualcuno dei tuoi vorrà sapere cosa sia;
> lascia che sia la grazia a darsi da fare,
> se vuole; lei che è insidiosa, è scaltra,
> che è sempre inattesa, è ostinata;
> che se non viene diritta, certo, trova sempre un qualche modo,
> per quanto strano, d'arrivare alla meta.
> Signor tenente, tu, devi solo partire,
> senza chiedere niente (p. 21).

15 agosto 1914, Messa dell'Assunzione nella Chiesa di Loupmont. Il tenente Péguy trova tra i suoi soldati un francescano:

> Qui, tenente Péguy, hai trovato un amico,
> uno dei soldati migliori – dici – della tua compagnia;
> un giovane frate, un cappuccino accorso dall'Italia, subito, i sandali ai piedi,
> all'annuncio della guerra.
> È magro, sofferente, il volto teso da un dolore silenzioso;
> sempre grinzoso.
> L'hai fatto cappellano dei soldati e tuo personale.
> Forse è stato lui stesso, oggi, per la festa dell'Assunzione, lui, che ti ha convinto
> a partecipare di nuovo,
> da quando eri bambino,
> alla messa, a confessarti;
> a ricevere un'assoluzione speciale,
> il corpo di Nostro Signore;
> a farti corpo, infine
> della Sua Chiesa (p. 70-71).

Non abbiamo documenti su questa probabile confessione e su questo ritorno ufficiale alla Chiesa, ma abbiamo una testimonianza di Raïssa Maritain che cita una lettera di Péguy alla sorella di Jacques Maritain con la quale lo scrittore aveva una grande confidenza. «Non andò mai a messa prima della guerra, prima del 15 agosto 1014; in quel giorno senza dubbi noi vi furono difficoltà per lui, in lui. Il 16 effettivamente scrisse a Jeanne Garnier-Maritain: "Forse un giorno vi dirò in quale parrocchia ho ascoltato la Messa dell'Assunzione. Se non ritornerò, andrete a Chartres una volta all'anno per me". Durante questa messa dell'Assunzione forse egli si comunicò. Ebbe la possibilità di confessarsi presso quel giovane cappuccino, uno dei soldati della compagnia, di cui egli parla, elogiandolo, in una lettera del 17 agosto 1914, aggiungendo: "Lo nomino assistente della mia compagnia e mio cappellano particolare". Ma tutto questo doveva-mo saperlo solo dopo la morte del nostro amico; tutto questo doveva maturare solo nella solitudine dei suoi ultimi giorni, nell'entusiasmo del sacrificio totale accettato». Raïssa Maritain aggiunge anche altre informazioni sulla morte dello scrittore, citando una lettera a Charlotte Péguy: «Un mese più tardi Claude Casimir-Périer, che aveva preso il comando della compagnia, scriveva a sua moglie: "Péguy la sera del 5 settembre sembrava presentire la sua fine gloriosa. Tutti quelli che l'hanno avvicinato l'hanno sentito come me... la vigilia aveva stazionato con i suoi uomini in un vecchio convento e passato la notte ad accumulare fiori ai piedi dell'altare della Vergine"».

5 settembre 1914, arrivo a Montmélian.

> Signor tenente l'hai trovata;
> come un caso,
> una fortuna sulla strada
> o come Dio, forse, ti ha promesso.
> Piccola chiesetta, oggi
> l'iscrizione che porti è diversa.
> Racconta di un tenente di Francia,
> che due giorni prima di morire, prima
> della sua battaglia, della prova che aspettava,
> trascorse la sua notte, qui, a raccogliere dei fiori
> e a sistemarli con pazienza

FIGURA 10: Altre pagine di *Prospettiva Persona* prodotte con LaTeX.

complessi sono stati risolti come e meglio che nel prodotto originale. I cambî di geometria vengono gestiti direttamente da LaTeX senza che il compositore debba selezionare a mano alcunché (eccetto un punto di interruzione del testo prima di un cambio di pagina e di geometria). La compilazione dell'indice avviene in maniera automatica, a differenza di quella manuale del prodotto originale. Si perde un po' nella flessibilità di posizionamento delle immagini, ma basterebbe giocare con le forme dei paragrafi per ovviare a questa "debolezza", e non è detto che un prossimo upgrade non preveda proprio questa possibilità. I punti di forza di LaTeX, inoltre, imprimono velocità e coerenza di produzione: i titoli, i posizionamenti, le voci bibliografiche, i cambiamenti linguistici e tutti gli altri automatismi proprî di LaTeX permettono risultati che i prodotti concorrenti faticano a ottenere.

Tutto questo concorre a realizzare una classe per la composizione di una rivista complessa come *Prospettiva Persona*.

La griglia del vicino è sempre più verde?

Abbiamo visto una serie di tecniche per convincere LaTeX a comporre i documenti nel rispetto di una griglia. Saremmo avvantaggiati usando LuaTeX o ConTeXt? Partiamo da quest'ultimo, tenendo conto che "non è la mia tazza di tè" (come non lo è LuaTeX) e che Luigi Scarso, che ritengo l'autorità italiana di ConTeXt, potrebbe scriverne in maniera circostanziata.

In Hagen (2003) Hans Hagen mostra la potenza degli *insiemi di colonne* (*column set*) per comporre semiautomaticamente le riviste. Ammetto che quello che vedo è sbalorditivo: figure su più colonne; riquadri su più colonne di pagine adiacenti; titolo su due pagine adiacenti; pagine con colonne in numero diverso o di diverse larghezze. In ognuno di questi casi, il testo mantiene sempre l'allineamento. È immediato pensare che avrei risparmiato qualche ora di lavoro se avessi dedicato almeno centinaia di ore a studiare ConTeXt. A ulteriore riprova della bontà di ConTeXt in questo campo, anche il capitolo 8 di Hagen (2002) mostra ottimi risultati, sebbene solo per documenti a singola colonna.

Non conoscendo affatto LuaTeX, ho provato a cercare in rete dei riferimenti sulla composizione a griglia con questo strumento. A oggi (inizio di marzo 2017) non ho reperito alcunché. Spulciando il LuaTeX Reference (LuaTeX development team, 2017), mi sembra di ravvisare delle facilitazioni grazie alla varietà dei *nodes* di LuaTeX e dei relativi attributi. Purtroppo lo stato delle mie competenze e conoscenze è ancora troppo limitato per permettermi più di una banale e disinformata congettura e serve solo a farmi desiderare di investigare ulteriormente.

Ringraziamenti

Un sentito grazie va a Giovanni Marcotullio, il redattore di *Prospettiva Persona* "innamorato" di LaTeX, che ha convinto i suoi direttori a tentare il "grande cambiamento".

Un ringraziamento particolare va a Jerónimo Leal, che ha indirizzato il suo ex allievo a me per la mia passata esperienza con Free Software Magazine (Pignalberi, 2005).

Grazie anche a chi, nella redazione di ArsTEXnica, mi ha dato suggerimenti utili a migliorare la qualità dell'articolo.

Altri ringraziamenti vanno agli autori dei pacchetti citati (e non inclusi in bibliografia) e a chi risponde alle innumerevoli domande su StackExchange. Molte delle domande sortemi per risolvere tutte le sfide avanzate da *Prospettiva Persona* erano già state poste. Non molto sorprendentemente, chi ha risposto in maniera esaustiva alla maggior parte di quelle domande è Enrico Gregorio. L'ultimo grazie va a lui.

Riferimenti bibliografici

Hagen, H. (2002). «It's in the details». URL http://www.pragma-ade.com/general/manuals/details.pdf.

— (2003). «Columns». URL www.pragma-ade.nl/general/manuals/columns.pdf.

LuaTeX development team (2017). «LuaTeX reference». URL www.luatex.org/svn/trunk/manual/luatex.pdf.

Pignalberi, G. (2015). «LaTeX in un'editrice universitaria: come e perché book non è adatta e una possibile alternativa funzionante». *ArsTEXnica*, (19), pp. 33–41. URL http://www.guitex.org/home/images/ArsTeXnica/AT019/editore.pdf.

— (22 ottobre 2005). «Della produzione di una rivista in LaTeX». In *Atti del secondo convegno nazionale di TEX, LaTeX e tipografia digitale*. Aula Magna, Scuola Superiore Sant'Anna, Pisa, pp. 25–56. URL http://www.guitex.org/home/images/meeting2005/articoli/pignalberi.pdf.

▷ Gianluca Pignalberi
g dot pignalberi at alice dot it

Foreign Function Interface in LuaTEX

Hans Hagen, Luigi Scarso

Abstract

We present a new Foreign Function Interface module for LuaTEX with the same API as `ffi` module. The paper shows some simple applications of a direct binding to a shared library and some tests done replacing the standard ConTEXt's Lua text shaper with Harfbuzz library's one. The results show that Harfbuzz performs better than Lua with some complex scripts like Arabic, while Lua is usually the best choice with a common Latin script. In that respect `jit` might come into the picture.

Sommario

Presentiamo un nuovo modulo Foreign Function Interface per LuaTEX con la stessa API di LuajitTEX. L'articolo illustra alcuni semplici casi di collegamento diretto con una libreria dinamica ed alcuni test fatti sostituendo il *text shaper* in Lua di ConTEXt con quello della libreria Harfbuzz. I risultati mostrano che Harfbuzz ha prestazioni migliori nel caso di alfabeti complessi come l'*arab*, mentre Lua è la scelta migliore per quelli latini. In questo contesto `jit` potrebbe avere un ruolo importante.

1 Introduction

The `libffi` library introduction (see https://github.com/libffi/libffi/tree/master/doc) gives the best explanation of what a 'Foreign Function Interface' is:

> "Compilers for high level languages generate code that follows certain conventions. These conventions are necessary, in part, for separate compilation to work. One such convention is the "calling convention". The calling convention is a set of assumptions made by the compiler about where function arguments will be found on entry to a function. A calling convention also specifies where the return value for a function is found. The calling convention is also sometimes called the "ABI" or "Application Binary Interface".
>
> Some programs may not know at the time of compilation what arguments are to be passed to a function. For instance, an interpreter may be told at run-time about the number and types of arguments used to call a given function. 'Libffi' can be used in such programs to provide a bridge from the interpreter program to compiled code.
>
> [...A] "libffi" library provides a portable, high level programming interface to various calling conventions. This allows a programmer to call any function specified by a call interface description at run time.
>
> FFI stands for Foreign Function Interface. A foreign function interface is the popular name for the interface that allows code written in one language to call code written in another language."

LuajitTEX provides the `ffi` module which offers in a very compact way almost the same functionality of `libffi` with the following remarkable differences:

- `ffi` supports less architecture/operating system pairs than `libffi`;
- `ffi` compiles and executes C declarations just in time, while `libffi` has no C parser.

Since version 1.0.3 (expected to appear in the mid of February 2017) even LuaTEX has a first implementation of `ffi` based on `luaffifb` (see https://github.com/facebook/luaffifb) that is still at an experimental stage. In the next sections we will show some applications of this module.

2 A first example

In the following code we will see how to directly use a C type. We declare a type `matrix` of size $N \times N$ as a linearly indexed array of `double`. Then we declare I as the identity matrix, $M : M[i, j] = (i + 1)/(j + 1)$, and perform $N = M \cdot I$, to check later that $N = M$.

```
\directluacode{
if  jit then
 --[==[ Luajittex ?  ]==]
 ffi = require("ffi")
 jit.on()
end
--[==[ ffi could be also disabled for this Arch/OS ]==]
if (type(ffi)=='table' and  ffi.os=='' and ffi.arch=='')
  then return
end
--[==[ ffi could be also fully disabled at runtime ]==]
if (type(ffi)=='table' and ffi.os==nil and ffi.arch==nil)
 then return
end
local C = ffi.C
```

```
--[==[ interface ]==]
ffi.cdef[[
  enum {N=256};
  typedef double matrix[N*N];
]]
local I = ffi.new('matrix')
for i=0,ffi.C.N-1 do I[i*C.N+i] = 1 end
local M = ffi.new('matrix')
for i=0,C.N-1 do
   for j=0,C.N-1 do M[j*C.N+i] = (i+1)/(j+1) end
end
local N = ffi.new('matrix')
for i=0, C.N-1 do
 for j=0, C.N-1 do
    N[j*C.N+i] = 0
    for k=0,C.N-1 do
     N[j*C.N+i] =
        M[j*C.N+k] * I[k*C.N+i]+N[j*C.N+i]
    end
 end
end
for i=0,C.N-1 do
   for j=0,C.N-1 do
     if M[j*C.N+i] ~= N[j*C.N+i] then
       print("TEST FAILED",
          M[j*C.N+i],N[j*C.N+i])
     end
    end
end
}
\end
```

First some words on the checks. As it is now, `ffi` is enabled if and only if `--shell-escape` is true and `--shell-restricted` is false (the `cnf` files are also considered) because `ffi` is inherently insecure (as we will discuss in greater detail later). Currently not all of the arch/OS pairs supported by LuajitTex are supported by `ffi`—practically only Intel x86_64 on Linux and OSX and Windows. However, support on Windows has some limitations due to a difference in underlying libraries. For normal usage this is not a problem. There is also a slight difference between LuaTEX, where `ffi` is globally available, and LuajitTEX, where it's only available on demand. The usual pattern is parsing a C *interface* and then using the declared types. In this example it's clear that the goal is to measure the time of pure computation and indeed `luajittex --luaonly` shows that the '`ffi`' version is about 1.8 times faster than the pure Lua one (on an Intel(R) Core(TM) i7-3610QM CPU @ 2.30GHz) with `jit.on()`) but the same code with `luatex --luaonly` is about 7.6 times *slower*. This is a first indication that there is still a lot of work to be done on the jit side.

The second example (courtesy of A. Kakuto) is more interesting: it shows how to call a function from the C run-time (also known as `libc` in Unix):

```
% courtesy of A. Kakuto
\documentclass{article}
\usepackage{luacode}
\begin{document}
\begin{luacode*}
--[==[ the usual ffi checks here .. ]==]
ffi.cdef[[
  typedef const char* LPCSTR;
  int system(LPCSTR lpcstr);
]]
local msvcr100 = ffi.load("msvcr100")
msvcr100.system("echo abc")
\end{luacode*}
\end{document}
```

Several remarks: first, the run-time is operating system-specific, so the code is not portable. Second, this is a way to bypass the protections against the execution of untrusted programs, so it's now clear the reason why `ffi` is only enabled with a full `shell-escape`, and even in this case it's possible to disable it by putting at the very beginning of the format input file the following Lua code:

```
ffi=require[[ffi]];
for k,_ in pairs(ffi) do
 if k~='gc' then
  ffi[k]=nil
 end;
end;
ffi=nil;
```

The following, more interesting example ends the section: the computation of the Fast Fourier Transform (FFT) of a sampled sequence. The library used is the `libfftw3.so` from http://www.fftw.org/, also available for Windows as `libfftw3-3.dll`:

```
if not(ffi) then
 ffi = require("ffi")
 jit.on()
end
local PI = math.pi
local sin= math.sin
local function sinc(t)
 if t==0 then
  return 1
 else
  return sin(PI*t)/(PI*t)
 end
end
-- For windows:
-- local fftw3 = ffi.load("./libfftw3-3.dll")
local fftw3 = ffi.load("./libfftw3.so")
ffi.cdef([[
  enum {
   FFTW_ESTIMATE = (1 << 6)
  };
```

```
  typedef struct fftw_plan_s *fftw_plan;
  typedef double complex  fftw_complex;
  void *fftw_malloc(size_t);
  fftw_plan fftw_plan_dft_r2c_1d(int n, double *in,
               fftw_complex *out, unsigned flags);
  void fftw_execute(const fftw_plan p);
  void fftw_destroy_plan(fftw_plan p);
  void fftw_free(void *p);
]])
local N = 1024*1024
local data_in  = ffi.new("double[?]", N)
local data_out = ffi.new("fftw_complex[?]",N/2+1)
local p,t
for i=0, N-1 do
  t = i/2
  data_in[i] = sinc(t)
end
p= fftw3.fftw_plan_dft_r2c_1d(N, data_in, data_out,
                              ffi.C.FFTW_ESTIMATE)
fftw3.fftw_execute(p)
local string_format = string.format
local real,imag,m,f
local mag = io.open('mag.txt','w')
local ph  = io.open('phase.txt','w')
for i=0, N/2 do
 real,imag = data_out[i].re,data_out[i].im
 m= math.sqrt(real^2+imag^2)/(PI/2)
 f= math.atan2(imag,real)
 mag:write(string_format("%f\n", m))
 ph:write(string_format("%f\n", f))
end
fftw3.fftw_destroy_plan(p)
```

Even if it's unlikely that users will do demanding runtime calculations when typesetting a document, computation is quite fast: for 1MByte of samples the timings are approximately 1.2 s / 2.1 s with LuaJitTeX (`jit.on()` / `jit.off()`) and 4.6 s for LuaTeX. The choice of the library is not accidental: the values of the FFT are `double complex`, but this type is not defined in the Microsoft C run-time so this code doesn't work; on the other hand, *it is* defined in the Windows MinGW `libgcc`, so the code works with MinGW. The solution could be `typedef double fftw_complex[2];` which is also accepted by `FFTW`, but this gives a segmentation fault in LuaTeX (while in LuaJitTeX is valid: again, something to fix for the 2018 TeX Live release).

The next section shows the probably most important application of `ffi`: the usage of the Harfbuzz library (`hb`) as text shaper (a text shaper is the function that converts Unicode text into glyph indices and positions). One of the author of this paper is the leading developer of ConTeXt. Based on research by Kai Eigner he has extended the format to enable a third party text shaper as a *plug-in*; it's hence possible to perform an effective comparison between the ConTeXt's native Lua text shaper and `hb` (for the record: this will be a module based option but it is not recommended for regular use as it has drawbacks). What follows is a verbatim excerpt from the documentation of this new functionality, the *Plug mode*, with the warning that `ffi` it's still at an experimental stage and some details may change.

3 Plug mode

During a past NTG meeting, Kai Eigner and Ivo Geradts demonstrated how to use the Harfbuzz (`hb`) library to process OpenType fonts. The main reason for playing with that was twofold: it would provide a way to compare the Lua-based font machinery against other methods and it could give a better performance for complex fonts and/or scripts.

One of the guiding principles of LuaTeX development is to provide no hard-coded solution. For that reason we opened up the internals so that advanced users may provide solutions written in pure Lua or can cooperate with libraries via Lua code as well. Hard-coded solutions make no sense as there is usually more than a possible solution to a specific problem, depending on one's need. Although development is closely related to ConTeXt, the development of the LuaTeX engine is independent and we try to be macro package-agnostic. Starting from a very early development stage we made the ConTeXt font handler compatible with other macro packages, though users might want to use a lighter one for special purposes. So we also kept the standard TeX font handler and called it `base` mode in ConTeXt, while the Lua handler is `node` mode because it operates on the node list. We will later refer to these modes by their names instead of their programming languages. Supporting `hb` as a way to compare the Lua-based font machinery against other methods is not a strong reason: we already have XeTeX as a milestone and, just in case we want to do a comparison against the standard, we have MS-Word.

The second reason (the look for better performance with complex fonts and/or scripts) could be significant for users because at least in a case the Lua variant performs better than the standard. Some fonts use many lookup steps or are even inefficient in using the available features. Anyway, up to now I haven't heard ConTeXt users complaining about speed. In fact, font handling became much faster during the latest few years, though probably no one noticed it. When using alternatives to the built-in methods, it is highly probable to lose some of the functionalities built into the current font system and/or its interactions with other mechanisms.

Just kicking in some alternative machinery is not the whole story. We still need to deal with the way TeX sees text: a sequence of glyph nodes, mixed

with discretionary nodes for languages that hyphenate, glue, kern, boxes, math, and more. Discretionary nodes are those text elements most difficult to manage. In contextual analysis, as well as positioning, we need to process up to three extras: the text preceding the node, the text following the node and the text to replace the current with, along with the occasional links to the preceding and/or following nodes. In case the font has features and the user activates one or more of them, we have to process again all of those subsequences, keeping an eye on spaces, as they can be involved in lookups, and on glyphs injection or deletion, that can add or remove one or more attributes.

Kai and Ivo are plain TEX users so they use a font definition and switching environment that is quite different from ConTEXt. In a usual ConTEXt-run the time spent on font processing is measurable but it's not the main bottleneck because other time consuming operations get executed. Sometimes, the load on the font subsystem can be higher because we provide additional features normally not found in OpenType. Add to that a more dynamic font model, and it will be clear that comparing performance between situations that use different macro packages is not that trivial (and relevant).

Another reason why we follow a Lua route is that we support (run time generated) virtual fonts, we are able to kick in additional features, we can let the font mechanism cooperate with other functionalities, and so on. And the current mechanisms are likely to acquire more trickery in the near future. We also need access to various font pieces of information. Since we had to figure out a lot of these OpenType things a decade ago, when standards were fuzzy, we allowed some tracing and visualisation.

After his presentation, Kai wrote an article and that was the moment that I looked into the code and tried to replicate his experiments. Since we're talking of libraries, it's clear that the topic is all but trivial, especially because I'm on another platform than he is: Windows instead of OSX. The first thing that I have done was rewriting the code that connects the library to TEX in a more suitable way for ConTEXt. Mixing with existing modes (base or node mode) makes no sense and it is asking for unwanted interferences, so I preferred to write a new plug mode. A sort of general text filtering mechanism was derived from the original code so that we can plug in whatever we want. After all, stability is not the strongest point of contemporary software development so when we depend on a library, we need to be prepared to switch to other (library based) solutions too (for instance, if I understood correctly, XETEX switched a few times).

After redoing the code the next step was to get the library running and that somehow failed due to some expected functions not being supported. At that time I thought it was a matter of interfacing. But, I could get around it by piping into the command line tools that come with the library and that was good enough for testing, although of course it was deadly slow. After that I just quit and moved on to something else.

Just before the publication of Kai's articles I tried the old code again and, surprise, after some messing around, the library worked. On my system the one shipped with Inkscape is used which is okay as it frees me from bothering about installations. I must admit that we have no real reason in ConTEXt for using fonts libraries but the interesting part was that it allowed me to play with the so called `ffi` interface of LuajitTEX. And, since that generates a nasty dependency, after a while Luigi Scarso and I managed to get a similar library working in stock LuaTEX (as that is the reference). So, I decided to give it a second try and in the process rewrote the interfacing code. After all, there is no reason for libraries not to be well-written and to have an optimised interface where possible.

Now, after a decade of writing Lua code, I dare to claim that I know a bit how to write relatively fast code so I was surprised to see that where Kai claimed that the library was faster than the Lua code, I saw that it really depends on the font. Sometimes the library approach is actually slower, which is not what I might expect. But, remember that complex fonts and scripts are a reason to use a library. What does 'complex' mean?

Most Latin fonts are not complex: ligatures and kerns and maybe a little bit of contextual analysis. Here the Lua variant is the clear winner. It runs up to ten times faster than the competitors. For more complex Latin fonts, like EBGaramond, which resolves ligatures in a different way, the library almost catches up, though the Lua handler keeps running faster. Keep in mind that we need to juggle discretionary nodes in any case. One difference between both methods is that the Lua handler runs over the whole lists (although it has to jump over fonts not being processed then) while the library gets snippets. However, tests show that the overhead involved in that is close to zero and can be neglected. Moreover, already long ago we have seen that when we compare MkIV LuaTEX and MkII XETEX, the Lua based font handler is not that slow at all, which makes sense because the problem doesn't change and, maybe more important, Lua is a pretty fast language. If one or the other approach is less that two times faster, the gain will probably go unnoticed in real runs. In my experience a few bad choices in macro or style writing is more harmful than a bit slower font machinery. Indeed, if you add one more node processing step, you will not notice a measurable slow down. By the way, one reason why font handling has been sped up over the years is that

our workflows sometimes have a high load and, for instance, remotely processing a set of 5 documents has to be fast. Also, in an edit workflow you want the runtime to be a bit comfortable.

Unlike Latin, a pure Arabic text has (normally) no discretionary nodes and the library profits most of that. I have to pick up the thread with Idris about the potential use of discretionary nodes in Arabic typesetting. On the contrary, Latin text has not so many replacements and positioning and therefore the LUA variant gets the advantage. Some of the additional features that the LUA variant provides can of course be provided for the library variant by adding some list pre- and post-processing but then you quickly lose any gain a library provides.

Kai's prototype has some cheats for right2left and special scripts like Devanagari. As these tweaks mostly involve discretionary nodes, there is no real need for them: when we don't hyphenate no time is wasted anyway. I didn't test Devanagari but there is some preprocessing needed in the LUA variant (provided by Kai and Ivo) that I might rewrite from scratch once I understand what happens there. I expect the library to perform somewhat better there. Eventually, I might add support for some more scripts that demand special treatments but so far I had no requests for it.

So, how about non-Latin? An experiment with Arabic—using the frequently used arabtype font—shows that the library performs faster, but when we use a mixed Latin and Arabic document the differences become less significant.

How did we measure? The baseline measurement is the so-called none mode: nothing is done there. It's fast but still takes a bit of time as it is triggered by a general mode identifying pass. That pass determines what font processing modes are needed for a list. Base mode only makes sense for Latin and has some limitations. It's fast and basically its run time can be neglected. That's why, for instance, PDFTEX is faster than the other engines, but it doesn't do Unicode well. NODE mode is the fancy name for the LUA font handler. The listed modes are ordered according to increasing run time. If we compare node mode with plug mode (in our case using the hb library), we can subtract none mode. This gives a cleaner (more distinctive) comparison but not a real honest one because there's always the need for the identifying pass.

We also performed a test with and without hyphenation but in practice that makes no sense: only verbatim text is typeset that way and normally we typeset that in none mode anyway. On the other hand, mixing fonts does happen. All of the tests start with forced garbage collection in order to get rid of that variance. We also pack into horizontal boxes so that the par builder (with all kind of associated callbacks doesn't kick in, although the node mode should compensate that).

The timings for LUATEX are the following.

luatex latin:

modern	t	$t-t_n$	$t-t_d$	t/t_d	$\frac{t-t_n}{t_d-t_n}$
context base	0.50	0.07	-0.82	0.38	0.08
context node	1.32	0.88	0.00	1.00	1.00
context none	0.43	0.00	-0.88	0.33	0.00
harfbuzz native	5.20	4.77	3.89	3.95	5.40

pagella	t	$t-t_n$	$t-t_d$	t/t_d	$\frac{t-t_n}{t_d-t_n}$
context base	0.54	0.07	-0.85	0.39	0.08
context node	1.39	0.92	0.00	1.00	1.00
context none	0.47	0.00	-0.92	0.34	0.00
harfbuzz native	5.16	4.69	3.77	3.72	5.12

dejavu	t	$t-t_n$	$t-t_d$	t/t_d	$\frac{t-t_n}{t_d-t_n}$
context base	0.49	0.07	-1.42	0.26	0.04
context node	1.91	1.48	0.00	1.00	1.00
context none	0.43	0.00	-1.48	0.22	0.00
harfbuzz native	5.25	4.82	3.34	2.75	3.25

cambria	t	$t-t_n$	$t-t_d$	t/t_d	$\frac{t-t_n}{t_d-t_n}$
context base	0.47	0.05	-1.86	0.20	0.03
context node	2.34	1.91	0.00	1.00	1.00
context none	0.43	0.00	-1.91	0.18	0.00
harfbuzz native	4.73	4.30	2.39	2.02	2.25

ebgaramond	t	$t-t_n$	$t-t_d$	t/t_d	$\frac{t-t_n}{t_d-t_n}$
context base	0.52	0.08	-3.44	0.13	0.02
context node	3.96	3.52	0.00	1.00	1.00
context none	0.44	0.00	-3.52	0.11	0.00
harfbuzz native	4.96	4.53	1.00	1.25	1.28

lucidaot	t	$t-t_n$	$t-t_d$	t/t_d	$\frac{t-t_n}{t_d-t_n}$
context base	0.50	0.02	-0.52	0.49	0.04
context node	1.02	0.54	0.00	1.00	1.00
context none	0.48	0.00	-0.54	0.47	0.00
harfbuzz native	4.47	3.99	3.45	4.40	7.41

luatex arabic:

arabtype	t	$t-t_n$	$t-t_d$	t/t_d	$\frac{t-t_n}{t_d-t_n}$
context base	0.47	0.01	-16.52	0.03	0.00
context node	16.99	16.53	0.00	1.00	1.00
context none	0.46	0.00	-16.53	0.03	0.00
harfbuzz native	6.88	6.42	-10.11	0.41	0.39

luatex mixed:

arabtype	t	$t-t_n$	$t-t_d$	t/t_d	$\frac{t-t_n}{t_d-t_n}$
context base	0.73	0.04	-8.02	0.08	0.00
context node	8.75	8.06	0.00	1.00	1.00
context none	0.69	0.00	-8.06	0.08	0.00
harfbuzz native	5.84	5.15	-2.91	0.67	0.64

The timings for LuajitTEX are of course overall better:

luajittex latin:

modern	t	$t-t_n$	$t-t_d$	t/t_d	$\frac{t-t_n}{t_d-t_n}$
context base	0.42	0.05	-0.42	0.50	0.11
context node	0.83	0.46	0.00	1.00	1.00
context none	0.37	0.00	-0.46	0.44	0.00
harfbuzz native	2.56	2.19	1.72	3.07	4.71

pagella	t	$t-t_n$	$t-t_d$	t/t_d	$\frac{t-t_n}{t_d-t_n}$
context base	0.46	0.05	-0.42	0.52	0.10
context node	0.87	0.46	0.00	1.00	1.00
context none	0.41	0.00	-0.46	0.47	0.00
harfbuzz native	2.48	2.08	1.61	2.85	4.48

dejavu	t	$t-t_n$	$t-t_d$	t/t_d	$\frac{t-t_n}{t_d-t_n}$
context base	0.40	0.04	-0.70	0.37	0.06
context node	1.10	0.74	0.00	1.00	1.00
context none	0.36	0.00	-0.74	0.33	0.00
harfbuzz native	2.52	2.16	1.42	2.29	2.91

cambria	t	$t-t_n$	$t-t_d$	t/t_d	$\frac{t-t_n}{t_d-t_n}$
context base	0.40	0.04	-0.92	0.30	0.04
context node	1.31	0.96	0.00	1.00	1.00
context none	0.36	0.00	-0.96	0.27	0.00
harfbuzz native	2.48	2.13	1.17	1.89	2.22

ebgaramond	t	$t-t_n$	$t-t_d$	t/t_d	$\frac{t-t_n}{t_d-t_n}$
context base	0.43	0.05	-1.76	0.20	0.03
context node	2.19	1.82	0.00	1.00	1.00
context none	0.38	0.00	-1.82	0.17	0.00
harfbuzz native	2.22	1.85	0.03	1.01	1.02

lucidaot	t	$t-t_n$	$t-t_d$	t/t_d	$\frac{t-t_n}{t_d-t_n}$
context base	0.42	0.01	-0.26	0.62	0.03
context node	0.67	0.27	0.00	1.00	1.00
context none	0.41	0.00	-0.27	0.61	0.00
harfbuzz native	2.28	1.87	1.61	3.39	7.06

luajittex Arabic:

arabtype	t	$t-t_n$	$t-t_d$	t/t_d	$\frac{t-t_n}{t_d-t_n}$
context base	0.36	0.01	-7.52	0.05	0.00
context node	7.88	7.53	0.00	1.00	1.00
context none	0.35	0.00	-7.53	0.04	0.00
harfbuzz native	2.15	1.80	-5.73	0.27	0.24

luajittex mixed:

arabtype	t	$t-t_n$	$t-t_d$	t/t_d	$\frac{t-t_n}{t_d-t_n}$
context base	0.61	0.03	-3.79	0.14	0.01
context node	4.40	3.83	0.00	1.00	1.00
context none	0.58	0.00	-3.83	0.13	0.00
harfbuzz native	2.54	1.97	-1.86	0.58	0.51

A few additional notes. Since a library is an abstraction, we need to make the best of it. In my case, I experienced a crash in `utf-32` mode. I could get around it but one advantage of using Lua is that it hardly crashes, (e.g., because as a scripting language it manages its memory well without user interference). My policy with libraries is just to wait till things get fixed and not to bother with the internals whys and hows.

Although ConTEXt will support the plug model, I won't officially use it (not even in documentation) so I can't support users in that. I didn't test the plug mode in real documents. Most documents I process contain Latin fonts or a mix; redefining feature sets or adapting styles for testing makes no sense. Now the question is: "can we just switch engine without looking at the way a font is defined?", the answer being: "not really, because (even with users not knowing about it) virtual fonts might be used or additional features be kicked in, so other mechanisms can make assumptions about how fonts are dealt with."

The usability of plug mode probably depends on the available workflow. We use ConTEXt in a few very specific workflows where interestingly we only use a small subset of its functionalities. Most of those workflows are user driven and tweaking fonts is popular and has resulted in all kind of mechanisms. Therefore it's unlikely that we will ever use it. If you process (in bursts) many documents in succession, each demanding a few runs, you don't want to sacrifice speed.

Of course timing can (and likely will) be different for plain TEX and LATEX usage. It depends on how mechanisms are hooked into the callbacks, what extra work is done or not done compared to ConTEXt. This means that my timings for ConTEXt will differ for sure from those of other packages. Timings are a snapshot anyway. And font processing is just one of the tasks to be executed. If you are not using ConTEXt you will probably use Kai's version because it is adapted to his use case and well tested.

A fundamental difference in the approach is that where the Lua variant operates on node lists only, the plug variant generates strings that get passed to a library (in the ConTEXt variant of `hb` support we use `utf-32` strings). It is interesting that a couple of years ago I considered using a similar method for Lua but eventually decided not to use it, first of all for performance reasons, but mostly because one still has to use some linked list model. I might pick up that idea as a variant but, since all this TEX related development doesn't really pay off and costs a lot of free time, it will probably never happen.

Using this mechanism (there might be variants in the future) allows the user to cook up special solutions. After all, that is what LuaTEX is about: the traditional core engine but with the ability to plug in your own code using Lua and this is just an example of it.

I'm not yet sure when the plugin mechanism will be in the ConTEXt distribution but it might happen once the `ffi` library is supported in LuaTEX.

At the end of this document we show the basics of the test setup, just in case you wonder what the numbers apply to.

Just to put things in perspective: the current (February 2017) MetaFun manual has 424 pages. It takes LuaTEX 18.3 seconds and LuajitTEX 14.4 seconds on my Dell 7600 laptop with 3840QM mobile i7 processor. It takes 6.1 (4.5) seconds of the total time to process 2170 ∓ graphics. Loading the 15 used fonts takes 0.25 (0.3) seconds including loading the outline of some. Font handling is part of the so-called hlist processing and takes around 1 (0.5) second and attribute backend processing takes 0.7 (0.3) seconds. One problem in these timings is that font processing often runs too fast to elapse it, especially when we have lots of small snippets. For example, short runs like titles and such simple texts take no time and verbatim needs no font processing. The difference in runtime between LuaTEX and LuajitTEX is significant so we can safely assume that we spend some more time on fonts than reported. Even if we add a few seconds, in this rather complete document, the time spent on fonts is still not that impressive, but a 5 times slower processing (we use mostly Pagella and Dejavu) would definitely add significantly to the total run time, especially if you need a few runs to get cross referencing etc. right.

4 Conclusion

Nonetheless, at this stage the LuaTEX `ffi` module does not offer the same performance, robustness and number of functions of the corresponding LuajitTEX module because it's still at an early stage of development.

Using native C types as a replacement of Lua types is not yet competitive as in LuajitTEX, and even in the latter environment the `jit` must be carefully managed to avoid to slow down the rest of the run.

The first results of using the direct binding to native shared libraries are encouraging and this can be useful in high specialised workflows. Nonetheless, interfacing with shared libraries also puts serious limits to the code portability and durability (i.e., the property of being easily processed by different releases of the same program) of the code.

Moreover, the main advantage over the corresponding SwigLib approach (see http://swiglib.foundry.supelec.fr) is to avoid the intermediate compilation step of the interface layer, saving another shared library and hence reducing the dependencies, though at the price of reducing the range of the supported architectures/operating systems and using a less sophisticated C parser than that implemented by SwigLib.

▷ Hans Hagen
Pragma ADE
The Netherlands
`http://luatex.org`

▷ Luigi Scarso
Padova, Italy
`luigi.scarso@gmail.com`
`http://luatex.org`

Questa rivista è stata prodotta
dal Gruppo Utilizzatori Italiani di TEX
usando esclusivamente software libero.

Versione elettronica per la diffusione via web.

Quattordicesimo convegno nazionale su TEX, LATEX e tipografia digitale

Call for Papers

Quest'anno il Convegno annuale su TEX, LATEX e tipografia digitale organizzato dal Gruppo Utilizzatori Italiani di TEX giunge al suo quattordicesimo appuntamento. Il Convegno sarà un momento di ritrovo e di confronto per la comunità LATEX italiana, tramite una serie di interventi atti sia a contribuire all'arricchimento sia a supportarne lo sviluppo.

Maggiori informazioni su data e luogo del Convegno e sulle modalità di presentazione degli interventi saranno disponibili all'indirizzo:

`http://www.guitex.org/guitmeeting/2017/`

ArsTEXnica

Rivista italiana di TEX e LATEX

Numero 23, Aprile 2017

3 Editoriale
Claudio Beccari

5 Comporre in griglia
Claudio Beccari

12 A Database Experiment With LuajitLATEX
Roberto Giacomelli

35 Storia della notazione musicale. Una ricognizione per immagini
Werner Lemberg

55 Riviste di aspetto complesso e composte a griglia con LATEX
Gianluca Pignalberi

70 Foreign Function Interface in LuaTEX
Hans Hagen, Luigi Scarso

9 771828 236001 70023