Utilizzo di caratteri TrueType con LaTeX
Un esempio pratico: i *Fell Types*

Massimiliano Dominici

Sommario

L'articolo mostra come è possibile sfruttare al meglio, con TeX, le potenzialità di un font TrueType. A questo scopo, viene esposta l'installazione di una collezione di font, i *Fell Types*, ricchi di caratteristiche non standard che TeX può essere istruito a gestire in maniera del tutto trasparente per l'utente.

Abstract

This paper explains how TeX can make the best use of the features of a TrueType font. For this purpose, the paper shows the installation of a collection of fonts, the *Fell Types*, full of non standard features that TeX can be taught to manage in a transperent way for the user.

1 Introduzione

Uno dei miti più duri da sfatare, riguardo a TeX, è quello secondo cui può essere utilizzato con un solo tipo di carattere[1] (il *Computer Modern*, naturalmente). Questa leggenda ha un suo fondo di verità. TeX ha potuto utilizzare, per molti anni, solo font in formato Metafont, dei quali non esisteva certo un'ampia scelta. E anche quando è stato possibile utilizzare font Type1 o TrueType, l'installazione di tipi di carattere diversi da quelli già presenti nella distribuzione usata ha continuato ad essere, a causa della complessità delle operazioni necessarie e della scarsa documentazione sull'argomento,[2] qualcosa che l'utente medio percepiva come al di sopra delle proprie capacità. Questa complessità, se da una parte implica anche un certo grado di complicazione, consente però di sfruttare le caratteristiche dei font ad un livello raggiunto dagli altri programmi di composizione tipografica solo con l'introduzione degli OpenType.

Oggi la diffusione sempre maggiore di quest'ultimo formato di font e lo sviluppo di estensioni di TeX (XeTeX e LuaTeX[3]) in grado di accedere direttamente ai font residenti nel sistema, senza dover passare attraverso i file ausiliari richiesti dall'implementazione di TeX, rende sicuramente più semplice l'utilizzo di nuovi tipi di carattere. Tuttavia, conoscere le potenzialità del vecchio sistema di accesso ai font (sistema che rimane comunque supportato anche dalle nuove implementazioni), consente di acquistare un controllo pieno sulle risorse disponibili, mettendo l'utente in grado di simulare le caratteristiche di un OpenType (sostituzione automatica di caratteri in base al contesto, ecc.) anche laddove queste non siano previste; oppure, nel caso si disponga di un OpenType, di "estendere" le funzionalità del font stesso al di là di quelle codificate al suo interno.

In questo articolo, dopo aver discusso, in generale, dell'installazione di un font TrueType, verrà mostrato un esempio concreto, scelto appositamente per la sua capacità di illustrare molte tra le manipolazioni possibili che TeX ci consente di operare su un font; e soprattutto quelle meno consuete. La specificità del font preso in esame fa sì che i metodi impiegati non possano essere generalizzati e applicati, così come sono, ad altri font, che, in genere, mostreranno diversi requisiti e diverse esigenze. Tuttavia tali metodi possono essere utili come "ricetta" da adattare alle circostanze.

Naturalmente, questo articolo non può esaurire l'intera materia relativa all'utilizzo dei font con LaTeX, di cui, invece, si limita ad evidenziare alcuni aspetti particolari. Per uno studio più sistematico dell'argomento rimandiamo, oltre che al classico e immancabile *TeXbook*,[4] al *LaTeX Companion*,[5] all'*Introduzione all'arte di scrivere con LaTeX* di Claudio Beccari,[6] e all'articolo di Emanuele Zannarini e Emiliano Giovanni Vavassori *LaTeX e i font: installazione pratica*,[7] che contiene una descrizione dettagliata dei vari formati di font esaminati nel presente articolo e un utilissimo glossario delle numerose estensioni di file riguardanti il vasto mondo dei caratteri digitali. Per una guida pratica all'installazione di font con LaTeX, infine, consigliamo la già citata *Font installation Guide*[8] di Philipp Lehmann.

1. Qui e di seguito uso il termine "carattere", o la locuzione "tipo di carattere", in riferimento ad un insieme di simboli per la stampa con caratteristiche grafiche comuni, considerate in maniera indipendente dalla tecnologia usata per tracciarli: il "tipo di carattere" *Times*. Il termine "font" si riferisce sempre, invece, alla sua versione digitalizzata.
2. La situazione è cambiata da quando è apparsa l'ottima guida pratica di Philipp Lehmann (Lehmann, 2004), ma la prima versione pubblica risale solo all'ottobre 2002.
3. Per maggiori informazioni su questi due programmi, si consultino i siti internet dei rispettivi progetti: `http://scripts.sil.org/xetex` e `http://www.luatex.org/`.
4. Knuth (1984).
5. Mittelbach *et al.* (2004).
6. Beccari (2007).
7. Zannarini e Vavassori (2005).
8. Lehmann (2004).

2 LaTeX e i caratteri da stampa

2.1 Lo schema di selezione dei font

Per determinare procedure e file ausiliari richiesti per poter utilizzare un font in generale (e un font TrueType in particolare), è necessario conoscere il modo in cui i font sono organizzati internamente, nello schema di selezione proprio di LaTeX 2ε ("New Font Selection Scheme", NFSS; si veda a questo proposito LaTeX3 Project Team (2005)).

LaTeX classifica ogni font a partire da cinque attributi: codifica, famiglia, serie, forma, corpo. Ogni volta che nel documento si ha un cambiamento di font (compresa la definizione del font iniziale) è come se venisse eseguita la seguente serie di istruzioni:

```
\fontencoding{<encoding>}
\fontfamily{<family>}
\fontseries{<series>}
\fontshape{<shape>}
\fontsize{<size>}{<baseline>}
\selectfont
```

L'argomento delle precedenti istruzioni viene immagazzinato nei rispettivi comandi interni: `\f@encoding`, `\f@family`, `\f@series`, `\f@shape`, `\f@size`, `\f@baselineskip`. Sono per l'appunto questi valori che vengono utilizzati per determinare il font da caricare. Per prima cosa, LaTeX esamina la combinazione codifica/famiglia, ovvero il valore di `\f@encoding/\f@family`. Se questo valore non è stato in precedenza dichiarato, LaTeX cerca un file corrispondente, con estensione `.fd`.[9] Ad esempio, nel caso in cui incontrasse, per la prima volta, la combinazione "T1/ptm", LaTeX andrebbe alla ricerca del file `t1ptm.fd`.

All'interno di questi file devono trovarsi le istruzioni per associare i valori contenuti nei comandi visti sopra con un preciso file `.tfm`, ovvero un file *TeX Font Metric*, che è tutto ciò che serve a LaTeX per comporre la pagina. L'effettiva inclusione del carattere, sia esso metafont, Type1 o TrueType, come vedremo in seguito, viene eseguita dal driver di stampa (sezione 2.2).

I "font definition files" contengono dunque delle tabelle che associano una determinata combinazione di codifica, famiglia, serie, forma e corpo ad un unico file `.tfm`. Queste tabelle vengono costruite per mezzo di due comandi:

```
\DeclareFontFamily{<encoding>}{<family>}
                  {<loading-settings>}
\DeclareFontShape{<encoding>}{<family>}
                 {<series>}{<shape>}
                 {<loading-info>}
                 {<loading-settings>}
```

`\DeclareFontFamily` oltre a dichiarare l'esistenza di una data famiglia per una data codifica, esegue il contenuto del terzo argomento ogni volta che viene caricato un font con tali caratteristiche. Ad esempio, per prevenire la divisione in sillabe dei caratteri a spaziatura fissa, è possibile passare come terzo argomento di `\DeclareFontFamily` il codice `\hypenchar\font=-1`. Il più delle volte, però, questo argomento è vuoto.

L'associazione effettiva di un file `.tfm` con un carattere avente specifici valori, viene effettuata tramite il comando `\DeclareFontShape`. In particolare in ⟨*loading-info*⟩ sono contenute le informazioni necessarie per associare i vari corpi con i font esterni. La sintassi è complessa e ci limiteremo qui ad esaminare un paio di esempi, in cui rimarcare i tratti essenziali. Il primo è tratto dal file `t1lmr.fd`, che contiene le definizioni per il carattere *Latin Modern Roman*, nella codifica T1. Quella che segue è la definizione inerente alla forma corsiva ("it") nel peso medio ("m"), ed è quella che si ottiene in questo documento[10] con il codice `\textit{Latin Modern Roman}`.

```
\DeclareFontShape{T1}{lmr}{m}{it}%
     {<-7.5>    ec-lmri7
      <7.5-8.5> ec-lmri8
      <8.5-9.5> ec-lmri9
      <9.5-11>  ec-lmri10
      <11->     ec-lmri12
      }{}
```

Le espressioni tra parentesi angolari rappresentano una gamma di dimensioni. Poiché il *Latin Modern* viene distribuito in diversi corpi, a differenti dimensioni vengono associati file differenti. Al di sopra di 11pt viene sempre utilizzato il file per il corpo 12, naturalmente adeguatamente scalato. Quando, poco sopra, LaTeX ha incontrato il codice `\textit{Latin Modern}`, ha consultato la dichiarazione precedente e ha trovato che per la combinazione "T1/lmr/m/it/10" doveva essere caricato il file `ec-lmri10.tfm`, dove si trovano le metriche richieste.

Il codice che segue è invece tratto da `t1yfrak.fd`, che contiene le definizioni per una versione dei caratteri gotici disegnati da Yannis Haralambous, in codifica T1.

```
\DeclareFontFamily{T1}{yfrak}{
   \hyphenchar \font =127}
\DeclareFontShape{T1}{yfrak}{m}{n}{
   <-> tfrak
}{}
\DeclareFontShape{T1}{yfrak}{m}{it}{
   <-> tfrakls
}{}
\DeclareFontShape{T1}{yfrak}{m}{sl}{
   <-> tswab
}{}
\DeclareFontShape{T1}{yfrak}{b}{n}{
   <-> tgoth
}{}
\DeclareFontShape{T1}{yfrak}{bx}{n}{
  <->ssub * yfrak/b/n
}{}
```

9. Sta per "font definition".

10. Questo documento, infatti, carica nel suo preambolo il pacchetto fontenc con opzione T1, per utilizzare la codifica T1 e il pacchetto lm, per utilizzare il carattere *Latin Modern*.

In questo caso i font sono distribuiti in un unico corpo e la cosa più rimarchevole è che si tratta di una collezione di font, più che una famiglia vera e propria. Infatti, il disegno di Fraktur (`tfrak`), Schwabacher (`tswab`) e Textur (`ygoth`) ha dei legami più tenui rispetto a quello dei nostri, rispettivi, tondo, corsivo e neretto.[11] Ciò mostra come sia possibile definire per LaTeX una famiglia di caratteri basata su font arbitrari.[12] Un'ultima notazione merita l'ultima dichiarazione del codice precedente. Indica la sostituzione di una serie ("bold-extended","bx") con un'altra ("bold","b"). LaTeX ha un meccanismo interno per gestire la sostituzione dei font mancanti. Senza entrare nei dettagli possiamo dire che nel caso in questione, senza l'ultima dichiarazione, ogni volta che avesse trovato un comando `\textbf` o `\bfseries`, mappati su "bx", LaTeX avrebbe fatto ricorso alla forma "m", come surrogato.

2.2 Inclusione dei font

A questo punto LaTeX sa quale file `.tfm` deve usare per comporre la pagina. Non gli serve altro. È come se mettesse in ordine, nelle posizioni appropriate, una serie di scatole vuote delle dimensioni giuste per accogliere i caratteri. L'inclusione vera e propria del font viene fatta da uno dei driver di stampa e la procedura varia a seconda del tipo di formato che si vuole ottenere. In realtà, il termine "driver di stampa" è un po' impreciso, perché in questo caso parliamo di un programma che genera un file (PDF, POSTSCRIPT, ecc.), oppure disegna i caratteri sullo schermo (nel caso di un visualizzatore di DVI, per esempio).

Il formato tradizionale di uscita di TeX è il DVI (DeVice Independent), pensato come formato intermedio che potesse essere trasformato nel linguaggio proprio di una qualunque stampante. Tuttavia, nel corso degli anni sono stati sviluppati diversi programmi per visualizzare direttamente a schermo i file in questo formato (Yap per Windows, Xdvi e Kdvi per sistemi *nix). Questi visualizzatori sono in grado di rendere direttamente dei font in formato TYPE1, oppure trasformare in un formato "bitmapped" (PK, Packed Font, estensione `.pk`), cioè non vettoriale,[13] i font in formato METAFONT e, con l'ausilio di `ttf2pk`, TRUETYPE.

Il meccanismo è pressappoco il seguente:

1. Il programma cerca nel file `psfonts.map` una riga che associ il `.tfm` usato ad un font TYPE1;
2. Se non lo trova cerca nelle appropriate cartelle un file `.pk` alla risoluzione richiesta oppure lo genera a partire da un file in formato METAFONT;
3. Se neanche questo è disponibile consulta la mappa `ttfonts.map` e, nel caso trovi una voce corrispondente, attiva `ttf2pk`.
4. Se non riesce a trovare un font da usare produce un messaggio di errore.

Lo stesso meccanismo è utilizzato da `dvips`, per generare un file POSTSCRIPT, che può eventualmente essere trasformato in PDF per mezzo dell'utilità `ps2pdf`, distribuita insieme all'interprete di POSTSCRIPT Ghostscript.

Per ottenere come formato finale un PDF, però, esistono soluzioni alternative, che permettono l'inclusione diretta di font di tipo TRUETYPE.[14] L'utilizzo, infatti, di font "bitmapped" sotto forma di TYPE3 in un documento PDF è sconsigliato, perché il font potrebbe apparire sgranato su alcuni visualizzatori, per quanto in fase di stampa non vi siano differenze apprezzabili.

La prima soluzione alternativa è quella di utilizzare `dvipdfm`, la cui procedura di inclusione dei font è simile, in pratica, a quella di `dvips`, con la differenza, appunto, che può includere direttamente font TRUETYPE, senza dover ricorrere ai `.pk`, e, di conseguenza, non effettua il terzo punto della procedura illustrata in precedenza (ed effettua le proprie ricerche nel file `dvipdfm.map`).

La soluzione più naturale, però, per ottenere un PDF è `pdftex`, a meno che non si abbiano vincoli particolari, come la presenza di grafici composti con PSTricks, ad esempio, che sfrutta caratteristiche avanzate del POSTSCRIPT non disponibili tramite `pdftex`. Di fatto, con installazioni di TeX ragionevolmente recenti, l'eseguibile `tex`, che si usa quando si vuole ottenere un DVI, è un collegamento proprio a `pdftex` in modalità DVI.

Con `pdftex` non è necessario, per ottenere un risultato visualizzabile, ricorrere ad una procedura in due stadi, come quella vista prima (composizione della pagina e poi, in un secondo tempo, inclusione dei caratteri tramite driver di stampa). È il programma stesso a provvedere all'inclusione dei font, cercando le associazioni tra `.tfm` e font veri e propri nella mappa `pdftex.map`. Anche in

11. Storicamente, però, il corsivo si è evoluto in maniera indipendente dal tondo, prima di diventare una variante all'interno della stessa famiglia.

12. Interviene, o dovrebbe intervenire, però, il gusto a porre dei limiti a tale arbitrarietà. Ciò non toglie che disponendo di un singolo tondo sprovvisto di un adeguato corsivo si può fare in modo che LaTeX usi automaticamente il corsivo di un'altra famiglia, se la sua forma si armonizza bene con quella del tondo.

13. I font in formato "bitmap" sono rappresentati tramite una matrice di punti (o mappa di bit) che indica al dispositivo incaricato di visualizzarli o di stamparli quale porzione dello schermo o della carta riempire di colore e quale no. Il risultato è che questo tipo di font richiede implementazioni diverse per dispositivi a diversa risoluzione. Un font "vettoriale", invece, dove i singoli caratteri vengono definiti tramite espressioni matematiche rappresentanti curve, può essere impiegato a risoluzioni diverse senza alcuna controindicazione.

14. Vedremo però che anche l'approccio descritto in precedenza consente, con alcuni accorgimenti, l'inclusione diretta di font TRUETYPE (sezione 4.5).

questo caso, essendo l'inclusione di un font TrueType diretta, il programma non ha bisogno di ricorrere a `ttf2pk`.

2.3 Mappe

Abbiamo visto nella sezione precedente che, per associare ad un determinato `.tfm` il relativo Type1 o TrueType, i vari programmi che si occupano dell'inclusione dei font fanno ricorso ad una o più mappe. Queste mappe possono essere installate nel sistema o caricate opzionalmente. In particolare `pdftex` in modalità PDF mette a disposizione la primitiva `\pdfmapfile` che può essere usata nel preambolo di un documento. Gli altri programmi, invece, come `dvips` o `dvipdfm`, essendo dei "post-processori", vengono chiamati da riga di comando ed è possibile, allora, caricare una particolare mappa passandola con l'appropriata opzione.

Tutte queste mappe, tranne quella utilizzata da `ttf2pk`, hanno una sintassi simile, anche se non identica, modellata su quella usata da `dvips`. In tutte, nessuna esclusa, deve essere indicato obbligatoriamente il file `.tfm`. Il nome del file contenente i caratteri può anche non essere specificato, nel caso che non si volesse includerlo, magari perché si vuole effettuare l'inclusione in un altro stadio del processo; in questo caso, però, deve essere indicato il nome "interno"[15] del font, in modo che il programma che, alla fine, dovrà fare l'inclusione, abbia un riferimento preciso.

In figura 1 è possibile esaminare la sintassi di questi file. È stata scelta a questo scopo la voce concernente il font LMRoman10-Regular, corrispondente al carattere *Latin Modern Roman* in corpo 10. Le righe relative a `psfonts.map` e `pdftex.map` sono identiche e mostrano nell'ordine: il nome del `.tfm`, il nome interno del font a cui si riferisce il `.tfm`, l'indicazione che il `.tfm` è stato ottenuto rimappando il file originale nella codifica (si veda, qui di seguito, la sezione 2.4) "enclmec", il file in cui tale codifica è specificata e, infine, il file in cui sono contenuti i caratteri. Le parentesi angolari che precedono il file di codifica e il Type1 indicano che tali file devono essere inclusi nell'output finale.[16]

La mappa usata da `dvipdfm` contiene invece un numero più stringato di informazioni, limitandosi al nome del `.tfm`, del vettore di codifica e del font.

Infine l'ultima voce è puramente ipotetica, dal momento che non è installato, di norma, nel sistema TeX un carattere *Latin Modern Roman* in formato TrueType. Si nota comunque che la sintassi è significativamente diversa da quella utilizzata per le altre mappe. Ricordiamo che questa mappa (`ttfonts.map`) viene utilizzata dai visualizzatori di DVI e da `dvips` per trasformare un TrueType in un font PK.

Sono state ovviamente trascurate molte delle opzioni che possono essere specificate in una mappa, e che indicano, ad esempio, le trasformazioni operate sul font per ottenere specifici effetti: se questo è stato inclinato, esteso, ecc. Per una visione più dettagliata della sintassi di questi file si possono consultare i manuali dei programmi `pdftex`, `dvips`, `dvipdfm` e `ttf2pk` inclusi nella propria distribuzione.

2.4 Vettori di codifica

Un vettore di codifica (in genere un file con estensione `.enc`) associa ad ogni singolo carattere contenuto in un dato font un valore numerico compreso tra 0 e 256. L'associazione avviene tramite il nome postscript del singolo carattere.[17] A tale nome corrisponde, nel font, una procedura che traccia il carattere in questione. In realtà, questo è strettamente vero solo per font di tipo POSTSCRIPT (Type1, Type3, ecc.), nei font TrueType i singoli caratteri sono identificati da un indice, invece che da un nome, ma, per questioni di compatibilità, esiste una tabella interna (`post`) che associa gli indici a nomi postscript.

All'interno di un file `.tfm` i singoli caratteri sono rappresentati appunto da valori numerici; il vettore di codifica traduce questi valori nei nomi delle procedure che il driver di stampa deve usare per tracciare, a schermo o sulla carta, i simboli corrispondenti.[18]

Ricodificare un font significa quindi associare agli stessi valori numerici caratteri (o meglio, simboli grafici) differenti. Questo può risultare utile (e lo è di fatto) per poter superare il limite di 256 caratteri accessibili da un singolo file `.tfm`.

2.5 Font virtuali

I font virtuali,[19] (*Virtual Fonts*, con estensione `.vf`) vengono menzionati qui per ultimi, ma rivestono un'importanza capitale se si vogliono sfruttare fino in fondo le potenzialità di un font. I font virtuali permettono, infatti, di operare sul font una serie di trasformazioni che vanno dalla ricodifica (si veda la sezione 2.4), alle alterazioni della forma (inclinazione, estensione, aggiunta di spazio addizionale, e in generale, qualsiasi trasformazione supportata dal linguaggio POSTSCRIPT), alla combinazione di caratteri provenienti da font reali differenti.

15. Il cosiddetto "nome postscript".

16. In particolare, la parentesi angolare singola posta davanti al Type1 significa che il font deve essere incluso solo parzialmente, cioè devono essere inclusi solo i caratteri effettivamente usati. Questo, oltre ad essere un modo per non appesantire eccessivamente il risultato finale, è spesso richiesto esplicitamente nella licenza dei font per evitare l'estrazione del font stesso dal documento. Due parentesi angolari avrebbero avuto il significato di inclusione completa.

17. È possibile vedere un esempio di vettore di codifica nella figura 2. L'associazione è implicita: al carattere in posizione 0, `/grave`, viene assegnato il valore numerico 0.

18. O, per usare un tecnicismo, i "glifi"; cioè i segni grafici che rappresentano concretamente i caratteri.

19. Per maggiori dettagli sull'argomento di questa sezione si veda Knuth (1990).

```
% Fonte: psfonts.map
ec-lmr10 LMRoman10-Regular "enclmec ReEncodeFont" <lm-ec.enc <lmr10.pfb

% Fonte: pdftex.map
ec-lmr10 LMRoman10-Regular "enclmec ReEncodeFont" <lm-ec.enc <lmr10.pfb

% Fonte: dvipdfm.map
ec-lmr10 lm-ec lmr10

% Fonte ipotetica: ttfonts.map
ec-lmr10 lmr10.ttf Encoding=lm-ec
```

FIGURA 1: Sintassi di un file `.map` per i diversi programmi.

Al suo interno un font virtuale indica i font reali da cui estrarre i singoli caratteri, i quali sono mappati su uno di questi (sotto forma di *TeX Font Metric* sprovvisto di *kerning* e legature; quello che comparirà nel file `.map`). Inoltre contiene tutte le informazioni (*kerning*, legature, trasformazioni su ogni singolo carattere) necessarie per rappresentare correttamente i caratteri.

TEX, però, non legge direttamente il font virtuale; abbiamo già visto che il suo compito consiste soltanto nel predisporre un insieme di scatole ben posizionate, pronte a ricevere, dal driver di stampa, i caratteri veri e propri. Quindi, le informazioni rilevanti per TEX (le metriche) sono duplicate in un file `.tfm`, compagno del font virtuale; ed è proprio questo file `.tfm` che viene letto da TEX in fase di composizione della pagina. Sarà il driver di stampa, leggendo nel font virtuale le trasformazioni da operare e l'associazione con i file `.tfm` "di base" (o "grezzi") rappresentati nelle mappe, a disegnare finalmente i caratteri nella forma voluta.

3 Installazione di un carattere TrueType

3.1 Panoramica generale

Ricapitolando quanto visto finora, ciò di cui abbiamo bisogno per poter utilizzare un font TRUETYPE con LATEX è:

1. Una serie di file *TEX Font Metric* `.tfm` che contengano le metriche del font (e, nel caso si vogliano sfruttare caratteristiche avanzate o effetti speciali, anche una serie di font virtuali `.vf`);

2. Una serie di file *Font Definition* `.fd` in cui ad una data combinazione di codifica/famiglia/serie/forma/corpo viene associato un particolare file `.tfm`;

3. Una serie di mappe in cui, ad un particolare file `.tfm` "di base" viene associato il corrispondente font;

4. Un vettore di codifica (`.enc`);

5. Infine un file di stile (`.sty`) per poter utilizzare convenientemente il font all'interno di un documento.

Le distribuzioni di LATEX attuali dispongono di una serie di utilità per gestire la creazione più o meno automatica di questi file. Le più comuni sono `ttf2tfm` e `fontinst`, che esamineremo nelle due sezioni seguenti. L'esame sarà, per forza di cose, sommario, e, per una conoscenza più dettagliata dei due programmi, si rimanda alla relativa documentazione.

3.2 Primo metodo: `ttf2tfm`

`ttf2tfm` viene distribuito insieme a `ttf2pk`. Si tratta di un programma da riga di comando con la seguente sintassi:

```
ttf2tfm ttffile[.ttf]
        [opzioni]
        [tfmfile[.tfm]]
```

Tramite le opzioni è possibile specificare la codifica del file "grezzo" e quella dell'eventuale font virtuale da generare, nonché applicare trasformazioni quali inclinazione o creazione di un finto maiuscoletto. L'esempio seguente servirà a dare un'idea più precisa del funzionamento:

```
ttf2tfm times.ttf -N -p 8r.enc \
-t T1-WGL4.enc -c .7 -V mtmrc8t.vpl \
-q mtmr8r.tfm >> ttfonts.map
```

In questo modo viene generato, a partire dal font TRUETYPE contenente Times New Roman, il file `.tfm` "grezzo" `mtmr8r.tfm`, nella codifica `8r`, nonché un file `.vpl`[20], da cui, mediante l'utilità `vptovf` è possibile ottenere il font virtuale `mtmrc8t.vf` e il suo compagno `mtmrc8t.tfm`. Questi ultimi rappresentano il maiuscoletto (finto) nella codifica `T1-WGL4`. L'opzione `-c` specifica di quanto vanno scalate le lettere maiuscole per ottenere il finto maiuscoletto, mentre l'opzione `-N` indica che, durante la ricodifica del font, bisogna accedere ai singoli caratteri mediante il loro nome postscript.

20. "Virtual Property List", è la versione in forma testuale dei file binari `.vf`.

Le informazioni normalmente mostrate sul terminale vengono redirette e aggiunte in coda al file `ttfonts.map`. L'opzione `-q` sopprime tutte le informazioni superflue, cosicché, alla fine, troveremo in fondo al suddetto file la seguente riga, che mappa il file `.tfm` "grezzo" `mtmr8r.tfm` sul file `times.ttf`:

```
mtmr8r times.ttf PS=Only Encoding=8r.enc
```

Il vettore di codifica "esterna", quello cioè utilizzato per creare il font virtuale, accetta una sintassi estesa rispetto a quella di un normale file `.enc`, che contiene solo una lista ordinata di 256 nomi di carattere. Le informazioni opzionali servono, ad esempio, ad istruire `ttf2tfm` su come comporre le legature richieste, oppure a sopprimere del *kerning*. In figura 2 è mostrato un estratto dal file `T1-WGL4.enc`. Da riga 27 a riga 31 si possono osservare le istruzioni per definire le legature; da riga 48 a riga 53 e poi da riga 124 a riga 131, rispettivamente, l'inizio e la fine della codifica vera e propria.

Un limite di `ttf2tfm` è che non è in grado di combinare insieme caratteri provenienti da font diversi. Questo, naturalmente, è un limite molto relativo per un programma pensato esplicitamente per operare su font TrueType in cui, di norma, tutti i caratteri, anche quelli appartenenti al cosidetto *expert set* (legature inusuali, maiuscoletto, numeri minuscoli, ecc.) quando sono presenti, si trovano inclusi nello stesso file. Tuttavia, questo malauguratamente non è sempre vero, e nell'esempio presentato più avanti (sezione 4) vedremo che saremo costretti ad assemblare caratteri da file diversi.

Un'altro limite è che il file di mappa ha una sintassi, come abbiamo visto anche in precedenza (si veda la figura 1), molto diversa da quella delle altre mappe e difficilmente traducibile (ad esempio non c'è modo di recuperare il nome postscript del file). Né `ttf2tfm` mette a disposizione alcuno strumento per generare mappe in un formato diverso.

3.3 Secondo metodo: `fontinst`

`fontinst` è un programma, completamente scritto in TeX, in grado di leggere un file di testo contenente istruzioni metriche in differenti formati (di norma un file Adobe Font Metric è un buon punto di partenza) e tradurre, dopo averle eventualmente modificate, queste istruzioni nel linguaggio tipico delle (*Virtual*) *Property List*. I file di testo `.pl` e `.vpl` creati da `fontinst` possono poi essere trasformati in `.tfm` e `.vf` per mezzo dei due programmi `pltotf` e `vptovf`, disponibili in ogni distribuzione di TeX. `fontinst`, inoltre, provvede alla creazione automatica dei file *Font Definition* (sezione 2.1) e semiautomatica delle mappe (sezione 2.3).

Un file `.tfm` "grezzo" può essere ottenuto, con `fontinst`, passando l'istruzione

\transformfont{*<tfm>*}{*<transforms>*}

L'argomento *<tfm>* è il nome del file da generare; all'interno dell'argomento *<transforms>* vanno indicate le operazioni da effettuare sul file stesso. Una delle più comuni è la ricodifica:

\reencodefont{*<encoding>*}{\fromafm{*<afm>*}}

Abbiamo specificato `\fromafm{`*`<afm>`*`}`, perché, il più delle volte si parte da un file `.afm`, ma altre istruzioni lecite avrebbero potuto essere `\frommtx{`*`<mtx>`*`}` o `\frompl{`*`<pl>`*`}`.

Un font virtuale può essere invece ottenuto con l'istruzione `\installfont`:

\installfont{*<vf>*}{*<mtx>*}{*<etx>*}
{*<encoding>*}{*<family>*}{*<series>*}
{*<shape>*}{*<size>*}

Il primo argomento è il nome del file `.vf`; gli argomenti da *<encoding>* in poi sono gli attributi del file, come verranno registrati nel file `.fd`. Le metriche e la codifica del file sono specificati, tramite appositi file ausiliari generati al volo oppure provvisti dal sistema o dall'utente, negli argomenti *<mtx>* e *<etx>*.

In particolare *<mtx>* provvede alla parte metrica vera e propria, specificando il *kerning* estratto dall'AFM e indicando quali caratteri è possibile simulare a partire da quelli già esistenti, e come, e quali no. Il *kerning* si trova di solito in un file con estensione `.mtx`, avente lo stesso nome di un `.tfm` "grezzo", e generato al volo da `fontinst` a partire da questo. La costruzione dei caratteri è invece contenuta in un file (sempre con estensione `.mtx`) provvisto dal sistema e dipendente dall'alfabeto usato. Nel caso dell'alfabeto latino si tratta di `newlatin.mtx`. L'istruzione `\unfakable{`*`<glyph>`*`}` indica che il carattere *<glyph>* è disponibile solo se esiste esplicitamente all'interno del font. In alcuni casi è possibile ottenere un carattere a partire da altri già esistenti. È il caso della legatura `fi` (l'esempio è tratto da `newlatin.mtx`):

```
\setglyph{fi}
   \glyph{f}{1000}
   \movert{\kerning{f}{i}}
   \glyph{i}{1000}
\endsetglyph
\setleftkerning{fi}{f}{1000}
\setrightkerning{fi}{i}{1000}
```

Il significato di questo codice è: "se non è disponibile un carattere `fi` distinto, usa il carattere `f`, poi spostati di uno spazio uguale al *kerning* tra `f` e `i`, quindi usa il carattere `i`; per finire, attribuisci a questo carattere a sinistra il *kerning* che avrebbe il carattere `f` e a destra quello che avrebbe il carattere `i`".

All'interno di un file `.etx`, da utilizzare nell'argomento *<etx>* di `\installfont`, si trova invece specificata la codifica del font virtuale. Ad ogni

```
 1  % T1-WGL4.enc
 2  %
 3  %
 4  % This is LaTeX T1 encoding for WGL4 encoded TrueType fonts
 5  % (e.g. from Windows 95)
27  % LIGKERN f i =: fi ;
28  % LIGKERN f l =: fl ;
29  % LIGKERN f f =: ff ;
30  % LIGKERN ff i =: ffi ;
31  % LIGKERN ff l =: ffl ;
48  /T1Encoding [          % now 256 chars follow
49  % 0x00
50    /grave /acute /circumflex /tilde
51    /dieresis /hungarumlaut /ring /caron
52    /breve /macron /dotaccent /cedilla
53    /ogonek /quotesinglbase /guilsinglleft /guilsinglright
124 % 0xF0
125   /eth /ntilde /ograve /oacute
126   /ocircumflex /otilde /odieresis /oe
127   /oslash /ugrave /uacute /ucircumflex
128   /udieresis /yacute /thorn /germandbls
129 ] def
130
131 % eof
```

Figura 2: Estratti dal vettore di codifica `T1-WGL4.enc`

carattere è assegnata una posizione e sono specificate inoltre le legature, sotto forma di tabella di sostituzioni. Quello che segue è un esempio tratto da `t1.etx`:

```
\setslot{\lclig{FF}{ff}}
  \ifnumber{\int{ligaturing}}>{0}\then
    \ligature{LIG}{%
      \lc{I}{i}}{\lclig{FFI}{ffi}
     }
    \ligature{LIG}{%
      \lc{L}{l}}{\lclig{FFL}{ffl}
     }
  \Fi
  \comment{The 'ff' ligature. It should
    be two characters wide in a monowidth
    font.}
\endsetslot
```

L'istruzione `\setslot` attribuisce al carattere `ff` (`\lclig` sta per "lowercase ligature", legatura minuscola, ed è definita in modo da assumere il valore del suo secondo argomento) un indice numerico, tramite un contatore progressivo, che ne stabilisce la posizione all'interno del vettore di codifica. Le istruzioni seguenti stabiliscono che il carattere in questione può formare una legatura se seguito dai caratteri `i` o `l`, (e specificatamente le legature `ffi` e `ffl`), ma solo se la variabile `ligaturing` è maggiore di zero. Questa possibilità di avere alcune caratteristiche legate ad un'espressione condizionale, rende estremamente semplice generare, a partire da uno stesso file `.etx`, diversi font virtuali che sfruttano diverse risorse del font reale.

Per ricapitolare, l'equivalente di quanto visto nella sezione 3.2 (creazione di un font virtuale contenente un falso maiuscoletto di Times New Roman, nella codifica T1) si può ottenere, quindi, scrivendo le seguenti righe in un file di testo e lanciando `tex` sul file stesso:

```
\input fontinst.sty
\setint{smallcapsscale}{700}
\recordtransforms{mtm-rec.tex}
\transformfont{mtmr8r}
  {\reencodefont{8r}{\fromafm{mtmr8a}}}
\installfonts
\installfamily{T1}{mtm}{}
\installfont{mtmrc8t}{mtmr8r,newlatin}{t1c}
  {T1}{mtm}{m}{sc}{}
\endinstallfonts
\endrecordtransforms
\bye
```

Alla fine della compilazione avremo un file *Property List* (`mtmr8r.pl`) da cui ricavare il corrispondente *TeX Metric File* "grezzo"; un font *Virtual Property list*, `mtmrc8t.vpl`, da cui ricavare il *Virtual Font* e il *TeX Font Metric* suo compagno; il file `t1mtm.fd`, con le associazioni tra gli attributi del font e i corrispettivi file metrici e il file `mtm-rec.tex` contenente le informazioni necessarie per generare mappe in vari formati.

Il seguente codice, una volta compilato con `tex`, permette di generare mappe per `pdftex`, `dvips` e `dvipdfm`:

```
\input finstmsc.sty
\adddriver{dvips}{mtm.map}
\adddriver{dvipdfm}{dvipdfm-mtm.map}
\input mtm-rec.tex
\donedrivers
\bye
```

Un'ultima notazione, infine. Tutto il procedimento seguito finora presuppone, al primo anello della catena, l'esistenza di un file AFM. I font TrueType, però, ne sono sprovvisti, in quanto, a differenza dei font Type1, le metriche sono immagazzinate

all'interno del font stesso, in apposite tabelle. Per "estrarre", quindi, un AFM da un TrueType, è necessario utilizzare il programma `ttf2afm`, che fa parte di una normale distribuzione di TeX, come nel seguente esempio:

```
ttf2afm -e 8a.enc -o mtm8a.afm times.ttf
```

È necessario specificare una codifica (opzione `-e`), altrimenti tutti i caratteri vengono mappati nella posizione `-1`, e l'AFM così generato è praticamente inservibile.

4 I *Fell Types*

4.1 I caratteri

I *Fell Types* sono una collezione di font digitalizzati da Igino Marini (`http://www.iginomarini.com/fell.html`), che riproducono alcuni tra i caratteri acquistati o commissionati alla fine del XVII secolo dal Dr John Fell per la tipografia dell'università di Oxford. La collezione comprende otto tipi di carattere, il cui nome indica, secondo le convenzioni dell'epoca, le dimensioni del corpo (si veda la tabella 1). Cinque di essi, utilizzabili per il corpo del testo, sono composti da tondo, corsivo e maiuscoletto; uno, per la titolazione, è privo delle minuscole e presenta come unica forma il tondo; e gli ultimi due offrono un centinaio scarso di ornamenti tipografici.

Il disegno dei caratteri, opera in gran parte dell'incisore olandese Peter de Walpergen, con il forte contrasto fra tratti fini e spessi e la relativa brevità degli ascendenti, è tipico della scuola olandese dell'epoca ed è uno dei primi esempi di transizionale barocco. Il carattere *English*, invece, dal disegno più raffinato (non a caso è stato preso a modello per le altre digitalizzazioni dei Fell Types, operate da *Hoefler & Frères* e *Dutch Type Library*) mostra connessioni più marcate con la tradizione francese e potrebbe essere opera di Christoffel Van Dijck, per quanto riguarda il tondo, e di Robert Granjon per il corsivo.

I caratteri originali comprendono, oltre ai caratteri usuali, alcune legature non consuete (`ct`, `st`, `fr`, ecc.), una variante della "s" minuscola (`longs`) e le legature di questa con altri caratteri, nonché varianti in posizione terminale o iniziale di alcune lettere. I numerali appaiono solo nella forma minuscola.

In fase di digitalizzazione sono stati aggiunti alcuni caratteri accentati e, soprattutto, un *kerning* molto accurato, ottenuto con il programma *iKern*, realizzato dallo stesso Igino Marini. Nessun font supera la soglia di 256 caratteri rappresentabili in un singolo vettore di codifica.

I *Fell Types* possono essere usati liberamente per la composizione di documenti, alla sola condizione di riprodurre nel documento stesso la nota di attribuzione "The Fell Types are digitally reproduced by Igino Marini. www.iginomarini.com"[21]. Attualmente, ad esempio, sono utilizzati nell'ambito del progetto *Electronic Texts in American Studies.*[22]

4.2 Preparazione dell'installazione

Prima di cominciare l'installazione dei caratteri[23], bisogna avere ben chiaro in mente il tipo di utilizzo che se ne intende fare. Possiamo pensare di usare i caratteri separatamente oppure come un'unica collezione, associando i singoli font a diverse dimensioni del corpo. Inoltre possiamo definire un'interfaccia che ci permetta di attivare a richiesta, come se avessimo a disposizione delle tabelle OpenType, caratteristiche particolari. Per esempio: attivare solo le legature tradizionali; attivare tutte le legature, più la "s lunga"; distinguere tra "s lunga" (all'inizio o all'interno di una parola) e "s" normale (alla fine di una parola).

L'interfaccia potrebbe essere ancora più complessa. Si potrebbe aggiungere l'uso automatico delle forme varianti. Si potrebbe scorporare l'uso della "s lunga" dall'uso delle legature, rendendola una caratteristica ortogonale alle altre. Tuttavia, questo, oltre ad aumentare il livello di complicazione, moltiplicherebbe a dismisura il numero dei font virtuali e metrici necessari a rappresentare il sistema. Ci limiteremo, quindi, ad un'interfaccia relativamente più semplice. I risultati che si ottengono attivando uno dei tre livelli di legature sono mostrati in figura 3, applicati alla prima frase dell'opera *Pseudodoxia Epidemica, or Vulgar Errors* di Thomas Browne, scrittore inglese della metà del XVII secolo.

The First and Father-cause of common Error, is, The common infirmity of Human Nature.

The Firſt and Father-cauſe of common Error, if, The common infirmity of Human Nature.

The First and Father-cauſe of common Error, is, The common infirmity of Human Nature.

FIGURA 3: Diversi livelli di legatura

Dal momento che sarà necessario avere un numero consistente di file *Font Definition*, relativi ai singoli tipi di carattere e alla collezione nel suo insieme, e di mappe, per i vari programmi; e che dovremo assemblare in un unico font virtuale caratteri provenienti da font diversi, la scelta del programma da utilizzare per l'installazione cade naturalmente su `fontinst`.

21. Le condizioni per la distribuzione sono più restrittive. Si possono consultare i termini della licenza sul sito dell'autore.

22. `http://digitalcommons.unl.edu/etas/`.

23. Il codice da cui sono tratti gli esempi di questa sezione e delle seguenti è consultabile per intero all'indirizzo `http://www.guit.sssup.it/downloads/felltypes.tar.gz`.

Nome	Incisore	Tondo	Corsivo	Maiuscoletto	Ornamenti	Corpo (pt)
Pica	De Walpergen	Sì	Sì	Sì	No	10.5
English	Van Dijck (?) – Granjon (?)	Sì	Sì	Sì	No	11.5
Great Primer	De Walpergen	Sì	Sì	Sì	No	14
Double Pica	De Walpergen	Sì	Sì	Sì	No	17
French Canon	De Walpergen	Sì	Sì	Sì	No	33
Three Line Pica	De Walpergen	Sì	No	No	No	41
Flowers1	Granjon et al.	No	No	No	Sì	25
Flowers2	Granjon et al.	No	No	No	Sì	17.5

TABELLA 1: I *Fell Types*

4.3 File ausiliari

Come abbiamo visto in precedenza (sezione 3.3) dobbiamo, per prima cosa, generare gli AFM richiesti da `fontinst`. Per fare questo, però, abbiamo bisogno di un vettore di codifica che non ci faccia perdere per strada qualche carattere.[24] L'ideale sarebbe usare, per ogni font, la codifica interna del font stesso. Come è possibile procurarsela, possibilmente con un procedimento automatizzabile? Un primo metodo consiste nel lanciare l'eseguibile `ttf2afm` sul font in questione, senza specificare il vettore di codifica. Nell'AFM compaiono, tutti mappati in posizione `-1`, i caratteri nell'ordine in cui vengono trovati. L'AFM può quindi essere modificato con un programma di manipolazione del testo (come `awk`, ad esempio, se si dispone di un sistema *nix), per ottenere il vettore di codifica nel formato voluto.

Alternativamente, si può fare ricorso a FONTTOOLS, un modulo Python reperibile all'indirizzo `http://sourceforge.net/projects/fonttools/`, che permette, tra le altre cose, di accedere ad una lista dei caratteri nella codifica propria del font. Basandosi su questo modulo e utilizzando il linguaggio Python, è possibile, appunto, trasformare questa lista in un file `.enc` sintatticamente corretto.

Successivamente, lanciando `ttf2afm` su ciascun file con l'appropriato vettore di codifica, si ottengono gli AFM voluti. Queste operazioni, che andrebbero ripetute su diciotto font, possono essere automatizzate abbastanza facilmente tramite uno script.

È invece, purtroppo, impossibile rendere automatica la creazione dei file `.mtx` e `.etx`. È possibile, però, utilizzare come base i corrispettivi file distribuiti con `fontinst`: `newlatin.mtx` e `t1.etx`.

Il file `newlatin.mtx` è strutturato in maniera modulare: le definizioni relative ai singoli caratteri sono distribuite nei file `llbuild.mtx`, `lubuild.mtx`, `ltpunct.mtx`, `lsfake.mtx`, `lsbuild.mtx`, `lsmisc.mtx`. È possibile, quindi, riutilizzare il codice contenuto nei singoli moduli, includendoli, mediante gli appropriati comandi, nel file `fell.mtx`, che potrà poi essere utilizzato nello script di `fontinst`. Non è necessario includere gli ultimi tre moduli sopra menzionati, in quanto essi contengono codice relativo al trattamento di font cosiddetti CAPS & SMALL CAPS, che usano per i caratteri in maiuscoletto nomi postscript come `Asmall`, ecc. Questo non avviene con i *Fell Types*, dove questi caratteri hanno gli stessi nomi dell'alfabeto minuscolo. Le definizioni relative ai caratteri non standard presenti nei *Fell Types* possono essere inserite, sotto forma di istruzioni `\setglyph` (vedi sezione 3.3), all'interno di un ulteriore modulo (`fell-specific.mtx`) da aggiungere agli altri. Si è scelto di utilizzare `\setglyph` piuttosto che `\unfakable` anche con caratteri non composti, in modo da avere un default cui ricorrere in mancanza di meglio. Così sarà possibile, utilizzando ad esempio l'*English*, usare il carattere `ealt` sapendo che, se si decide di ricomporre il documento in *Pica*, che non possiede tale carattere, in quel punto apparirà una `e` e non un quadratino nero.

Per quanto riguarda il file `.etx` (`t1fell.etx`), si può ottenere editando una copia di `t1.etx`, eliminando i riferimenti a caratteri che non compaiono nei *Fell Types* e sostituendoli con altri, oppure con *slot* vuoti. Può essere interessante esaminare il codice relativo alla lettera `s`, perché è un buon esempio di come si può controllare l'attivazione, in un font virtuale, di una data caratteristica.

```
\ifnumber{\int{ligaturing}}<{1}\then
\setslot{s}
  \comment{The ‘s’ character}
\endsetslot
\Else
\setslot{longs}
  \ligature{LIG}{h}{longsh}
  \ligature{LIG}{i}{longsi}
  \ligature{LIG}{l}{longsl}
  \ligature{LIG}{longs}{dbllongs}
  \ligature{LIG}{t}{longst}
  \ifnumber{\int{ligaturing}}>{1}\then
  \ligature{LIG}{boundary}{s}
  \ligature{LIG/}{quoteright}{s}
  \ligature{LIG/}{guilsinglleft}{s}
  \ligature{LIG/}{guilsinglright}{s}
  \ligature{LIG/}{quotedblleft}{s}
  \ligature{LIG/}{quotedblright}{s}
  \ligature{LIG/}{guillemotleft}{s}
  \ligature{LIG/}{guillemotright}{s}
  \ligature{LIG/}{period}{s}
  \ligature{LIG/}{comma}{s}
  \ligature{LIG/}{colon}{s}
```

24. Si richiama l'attenzione del lettore sul fatto, già accennato nella sezione 3.3, che ogni carattere presente nel font ma non contemplato dal vettore di codifica viene mappato in posizione `-1` e viene di conseguenza ignorato dai programmi che leggono l'AFM.

```
  \ligature{LIG/}{semicolon}{s}
  \ligature{LIG/}{rangedash}{s}
  \ligature{LIG/}{punctdash}{s}
  \ligature{LIG/}{exclam}{s}
  \ligature{LIG/}{question}{s}
  \ligature{LIG/}{slash}{s}
  \ligature{LIG/}{hyphen}{s}
  \ligature{LIG/}{parenleft}{s}
  \ligature{LIG/}{parenright}{s}
  \Fi
  \comment{The 'longs' character.}
\endsetslot
\Fi
```

Se la variabile `ligaturing` ha valore minore di 1, allora viene usata la "s" normale, altrimenti viene usata la variante "lunga" con tutte le sue legature. Ma, se la variabile è maggiore di 1 la "s lunga" viene di nuovo trasformata in "s" normale in fine di parola (ovvero quando è seguita da uno spazio bianco, `boundary`, o da punteggiatura). Il codice `LIG/` indica un tipo particolare di legatura, dove il secondo carattere che va a formare la legatura viene mantenuto. Più oltre, nello stesso file, ci si assicura che anche la legatura `longst` si comporti alla stessa maniera di `longs` e si trasformi in `st` in fine della parola.

Infine, poiché è conveniente costruire font virtuali e metrici a partire da una versione "ricodificata" del font, è necessario anche preparare due vettori di codifica che contengano, rispettivamente, tutti i caratteri presenti nella forma tondo e corsivo l'uno (`fell.enc`), e maiuscoletto l'altro (`fell-sc.enc`).[25]

4.4 File metrici e definizioni

Una volta concluse queste operazioni preliminari, è possibile preparare lo script che istruirà `fontinst` nel processo di creazione dei file metrici. Contestualmente verranno creati anche i file *Font Definition* e le mappe.

```
\input finstmsc.sty
\input fontinst.sty
\needsfontinstversion{1.927}
\enctoetx{fell}{fell}
\enctoetx{fell-sc}{fell-sc}
\recordtransforms{fell-rec.tex}
```

Dopo aver caricato `finstmsc.sty` e `fontinst.sty`, e aver dichiarato la versione di `fontinst` in uso, i due vettori di codifica `fell.enc` e `fell-sc.enc` vengono trasformati in file `.etx` tramite `\enctoetx`. Quindi viene attivata la registrazione delle operazioni sul file `fell-rec.tex`. Sarà a partire dalle informazioni trascritte automaticamente su questo file che verranno, in seguito, generate le mappe per i vari programmi.

I file metrici "grezzi" ricodificati si ottengono, come è stato mostrato in sezione 3.3, tramite il comando `\transformfont`. Nel codice seguente, tale operazione viene mostrata limitatamente al tondo, corsivo e maiuscoletto del tipo di carattere *Pica*.

```
\transformfont{picar}{%
  \reencodefont{fell}{\fromafm{FePIrm2}}}
\transformfont{picari}{%
  \reencodefont{fell}{\fromafm{FePIit2}}}
\transformfont{picarc}{%
  \reencodefont{fell-sc}{\fromafm{FePIsc2}}}
```

Generati i file metrici "grezzi", è necessario trasformarli in font virtuali e creare i *Font Definition* corrispondenti. Questo avviene all'interno della coppia di comandi `\installfonts`, `\endinstallfonts`. In precedenza, però, va attivato il livello di legatura voluto, con il comando `\setint{ligaturing}{<int>}`

```
\setint{ligaturing}{0}
\installfonts
\installfamily{T1}{fell0}{}
\installfont{ecanonr0}
  {ecanonr,ecanonrc,fell}{t1fell}
  {T1}{fell0}{m}{n}{<-10.95>}
\installfont{picar0}
  {picar,picarc,fell}{t1fell}
  {T1}{fell0}{m}{n}{<10.95-14.4>}
\installfont{greatpr0}
  {greatpr,greatprc,fell}{t1fell}
  {T1}{fell0}{m}{n}{<14.4-17.28>}
\installfont{dpicar0}
  {dpicar,dpicarc,fell}{t1fell}
  {T1}{fell0}{m}{n}{<17.28-24.88>}
\installfont{fcanonr0}
  {fcanonr,fcanonrc,fell}{t1fell}
  {T1}{fell0}{m}{n}{<24.88->}
\installfont{ecanonri0}
  {ecanonri,ecanonrc,fell}{t1fell}
  {T1}{fell0}{m}{it}{<-10.95>}
\installfont{picari0}
  {picari,picarc,fell}{t1fell}
  {T1}{fell0}{m}{it}{<10.95-14.4>}
\installfont{greatpri0}
  {greatpri,greatprc,fell}{t1fell}
  {T1}{fell0}{m}{it}{<14.4-17.28>}
\installfont{dpicari0}
  {dpicari,dpicarc,fell}{t1fell}
  {T1}{fell0}{m}{it}{<17.28-24.88>}
\installfont{fcanonri0}
  {fcanonri,fcanonrc,fell}{t1fell}
  {T1}{fell0}{m}{it}{<24.88->}
\installfont{ecanonrc0}
  {ecanonrc,fell}{t1fell}
  {T1}{fell0}{m}{sc}{<-10.95>}
\installfont{picarc0}
  {picarc,fell}{t1fell}
  {T1}{fell0}{m}{sc}{<10.95-14.4>}
\installfont{greatprc0}
  {greatprc,fell}{t1fell}
  {T1}{fell0}{m}{sc}{<14.4-17.28>}
\installfont{dpicarc0}
  {dpicarc,fell}{t1fell}
  {T1}{fell0}{m}{sc}{<17.28-24.88>}
\installfont{fcanonrc0}
  {fcanonrc,fell}{t1fell}
  {T1}{fell0}{m}{sc}{<24.88->}
\endinstallfonts
```

Il codice mostrato qui sopra illustra, per l'appunto, l'installazione della famiglia di font `fell0`, corrispondente all'intera collezione di font contenente

25. `fontinst` richiederebbe a questo scopo un file `.etx`, ma poiché è in grado di costruirselo da solo sulla base di un `.enc`, e quest'ultimo è molto più semplice da scrivere, il procedimento sopra indicato non ha conroindicazioni.

soltanto le legature usuali (variabile `ligaturing` uguale a 0). Il comando `\installfamily` è responsabile della scrittura, nel file `t1fell0.fd`, della riga `\DeclareFontFamily{T1}{fell0}{}` (si veda la sezione 2.1), mentre ogni chiamata a `\installfont` genera un file `.vpl` e contemporaneamente crea una voce nel file `t1fell0.fd` che associa il file metrico ad una determinata combinazione di attributi.

Il codice riguardante ogni singola istruzione `\installfont` è stato diviso su tre righe sia per esigenze di spazio, sia per separare anche visivamente i diversi compiti svolti dall'istruzione stessa. Nella prima riga è indicato il nome del file `.vpl` (e quindi del font virtuale e del *TeX Font Metric* che ne risulteranno), ad esempio `picar0`. Nella seconda sono indicati i file ausiliari necessari per generare il file `.vpl`, ovvero i file `.mtx` `picar`, `picarc` e `fell`, e il file `.etx` `t1fell`. Poiché all'interno del tondo e del corsivo mancano alcuni caratteri presenti nella codifica (ad esempio le parentesi graffe), è necessario recuperarle dal corrispondente maiuscoletto: da qui l'esigenza di usare `picarc` accanto a `picar`. Infine nella terza riga sono indicate le informazioni che verranno scritte nel file `t1fell0.fd` sotto forma di istruzioni `\DeclareFontShape`. Si può notare che ad ogni *TeX Font Metric* è associata soltanto una gamma di dimensioni, ad esempio `picar0` verrà usato solo per dimensioni che vanno da 10.95pt a 14.4pt.

Dal momento però che vogliamo poter usare i vari tipi di carattere anche singolarmente, è necessario installare le singole famiglie. Non è, tuttavia, necessario creare nuovi `.vpl`. Il codice che segue, e che va ripetuto per ogni famiglia di font, si limita a creare il file `t1pica0.fd`, associando le diverse combinazioni di attributi al *TeX Font Metric* indicato nel primo argomaento di `\installfontas`.

```
\installfonts
\installfamily{T1}{pica0}{}
\installfontas{picar0}{T1}{pica0}{m}{n}{}
\installfontas{picari0}{T1}{pica0}{m}{it}{}
\installfontas{picarc0}{T1}{pica0}{m}{sc}{}
\endinstallfonts
```

Queste operazioni vanno ripetute immutate, cambiando la variabile `ligaturing` e ponendola uguale prima a 1 e poi a 2, in modo da avere le famiglie di font `t1fell1` e `t1fell2`, `t1pica1` e `t1pica2`, ecc., relative agli ulteriori due livelli di legature.

Lo stesso procedimento è valido per installare le medesime famiglie di font nella codifica TS1, *Text Companion*, che contiene i simboli utilizzati in modalità non matematica. Una tipica istruzione `\installfont`, in questo caso, ha il seguente aspetto:

```
\installfont{tc-picar0}
  {picar,picarc,textcomp}{ts1}
  {TS1}{fell0}{m}{n}{<10.95-14.4>}
```

L'unica differenza rispetto all'installazione della famiglia in codifica T1, è rappresentata dall'uso del file `textcomp.mtx` al posto di `fell.mtx` e di `ts1.etx` al posto di `t1fell.etx`. Va sottolineato il fatto che non è necessario generare i font virtuali in codifica TS1 anche per il maiuscoletto, né per ogni livello di legature; basterà quindi utilizzare istruzioni `\installfontas` invece che `\installfont`.

Restano da installare ancora il tipo di carattere *Three Line Pica*, usato per la titolazione, e le due collezioni di ornamenti. Questi ultimi, essendo font contenenti caratteri non testuali, non hanno bisogno di ricorrere ad una codifica. Per indicare questo fatto, verrà usata la pseudo codifica U (*Unknown*).

```
\installfonts
\installfamily{U}{flow}{}
\installrawfont{FeFlow1}
  {FeFlow1}{txtfdmns,FeFlow1 mtxasetx}
  {U}{flow}{m}{n}{}
\installrawfont{FeFlow2}
  {FeFlow2}{txtfdmns,FeFlow2 mtxasetx}
  {U}{flow}{m}{sc}{}
\endinstallfonts
```

Inoltre, come si vede dal codice qui sopra, verrà usato direttamente il file `.tfm` "grezzo" (istruzione `\installrawfont`). Non essendo specificata alcuna codifica si indica a `fontinst` di dedurla dal file `.mtx` stesso (`mtxasetx`), aggiungendo, ad ogni buon conto, i parametri `\fontdimen` che servono a TEX per poter utilizzare il font (`txtfdmns`).

Three Line Pica, invece, pur essendo un tipo di carattere per il testo, deve essere utilizzato solo in particolari contesti, come titoli, capilettera, ecc. Inoltre non è influenzato in alcuna maniera dalla variabile `ligaturing`. Ciò significa che, una volta generato il corrispondente file metrico "grezzo", l'installazione è semplicissima:

```
\installfonts
\installfamily{T1}{tlpica}{}
\installfont{tlpicar}
  {tlpicaraw,fell}
  {t1fell}{T1}{tlpica}{m}{n}{}
\endinstallfonts
```

Lo script si conclude chiudendo il file `fell-rec.tex` e uscendo dal programma.

```
\endrecordtransforms
\bye
```

Alla fine della compilazione con `tex`, saranno presenti nella cartella di installazione un certo numero di file con estensione `.pl` e `.vpl`. Lanciando su questi file, rispettivamente, gli eseguibili `pltotf` e `vptovf`, si otterranno i font virtuali e i *TeX Font Metric*, da copiare nelle cartelle `fonts/vf/imfell/` e `fonts/tfm/imfell/`. I file *Font Definition* e i TrueType possono essere spostati, così come sono nelle cartelle `tex/latex/imfell/`

e `fonts/truetype/imfell/`.[26] Tutte queste operazioni possono essere automatizzate nello script di installazione.

4.5 Mappe

Come illustrato in precedenza (sezione 3.3), `fontinst` è in grado di generare mappe per i più comuni driver di stampa (`dvips`, `pdftex` e `dvipdfm`), con un procedimento in due fasi. Nella prima fase lo script utilizzato per installare i font genera un file in cui sono raccolte tutte le informazioni rilevanti. Nella seconda, questo file viene richiamato all'interno di un secondo script che si occupa specificamente di generare le mappe.

Sfortunatamente, se il font da installare è un TrueType, le cose si complicano. Innanzitutto deve essere generata anche una mappa per `ttf2pk`, in modo che i visualizzatori di DVI possano correttamente disegnare i font a schermo. In secondo luogo nelle mappe per `dvips` e `pdftex` deve essere indicato un vettore di codifica anche quando il generatore di mappe, pensato per font Type1, originalmente incluso in `fontinst` non lo farebbe (cioè quando il font non è stato ricodificato). Questo perché altrimenti `pdftex` non è in grado di effettuare l'inclusione parziale dei font.

Infine va considerato un ultimo fatto. `dvips` non è in grado di includere un font TrueType in un documento PostScript al volo sotto forma di font Type42.[27] Quindi, un documento realizzato con `dvips` contiene solo font bitmap generati a partire dai TrueType. Questo, specialmente se il documento deve essere ulteriormente distillato in PDF, può non essere un risultato auspicabile. Si può, allora, far ricorso a Ghostscript, il quale, invece, è in grado di fare l'inclusione al volo, sia per visualizzare il documento con GSView o altri visualizzatori, sia per distillare in PDF con `ps2pdf`. Per fare questo, però, è necessario che nella mappa per `dvips` non vengano specificati i font da includere, dimodoché il programma inserisce solo un riferimento e sarà poi l'inteprete di PostScript, cioè, nel caso in esame, Ghostscript, ad effettuare materialmente l'inclusione. Tutto questo significa che è necessario preparare due diverse mappe per `dvips` e `pdftex`.

Bisogna quindi scrivere, in un file che verrà chiamato `adddrivers.sty`, dei nuovi generatori di mappe, usando come base quelli forniti da `fontinst` nel file `finstmsc.sty` e modificandoli in modo da ottenere i risultati voluti. I nuovi generatori, a cui è bene attribuire un nome che ricordi che sono stati creati appositamente per gestire file TrueType, verranno poi richiamati all'interno dello script `fell-map.tex`, come si vede nel codice che segue.

```
\input finstmsc.sty
\input adddrivers.sty
\resetstr{PSfontsuffix}{.ttf}
\adddriver{ttfdvips}{dvips-fell.map}
\adddriver{ttfpdftex}{pdftex-fell.map}
\adddriver{ttfdvipdfm}{dvipdfm-fell.map}
\adddriver{ttftopk}{ttf2pk-fell.map}
\input fell-rec.tex
\donedrivers
\bye
```

Inoltre la variabile `PSfontsuffix` va posta uguale a `.ttf`, altrimenti `fontinst` utilizzerebbe il suffisso di default, che è `.pfa`.[28]

Le mappe così ottenute andranno copiate nelle cartelle dove possono essere trovate dai rispettivi programmi. Nello specifico: `fonts/map/dvips/`, `fonts/map/pdftex/`, `fonts/map/dvipdfm/` e `fonts/map/ttf2pk/`. Di norma, nel caso di un font Type1 la mappa è solo una, e il suo contenuto può essere automaticamente aggiunto, con le dovute modifiche, alle mappe globali lette dai vari programmi. Di solito esiste, nella propria distribuzione di TeX, una utilità (che nella TeX Live si chiama `updmap`) che si occupa proprio di questo.

Nel nostro caso questo non è possibile e ogni mappa va trattata separatamente. Se la compilazione viene effettuata tramite `pdftex` la mappa può essere caricata direttamente dal documento grazie alla primitiva `\pdfmapfile`. Se la compilazione è avvenuta con `tex` il visualizzatore DVI e `dvips` non trovando nessuna voce relativa ai *TeX Font Metric* all'interno della mappa globale cercheranno automaticamente in `ttfonts.map`, dove si sarà provveduto ad aggiungere il contenuto di `ttf2pk.map` (manualmente o tramite comando nello script di installazione). Infine, se si vuole ottenere un PDF con `dvipdfm` o con `dvips` seguito da `ps2pdf`, è necessario passare la mappa al compilatore tramite l'apposita opzione (`-f` per `dvipdfm` e `-u` per `dvips`).

4.6 File di stile

È consigliabile avere a disposizione una serie di istruzioni per non dover ricorrere a comandi di basso livello ogni volta che si vuole cambiare il tipo di carattere usato, o il suo livello di legature. Queste istruzioni vanno inserite in un apposito file di stile da caricare nel documento, con il consueto comando `\usepackage`. Qui di seguito non verrà illustrato l'intero contenuto del file di stile, ma solo alcuni tratti salienti. Si rimanda, come al solito, ai commenti del sorgente, per una spiegazione completa.

26. Si intende, qui e oltre, sempre a partire dalla radice `$TEXMF/`

27. Un font Type42 è sostanzialmente un font TrueType inglobato in un contenitore compatibile con il linguaggio PostScript.

28. Non entreremo qui nei dettagli di come scrivere un generatore di mappe. Come al solito, il lettore curioso può consultare il codice a disposizione all'indirizzo `http://www.guit.sssup.it/download/`.

Nel file di stile in questione dovrà essere possibile specificare il tipo di carattere e il livello di legature usati, sia come opzione del pacchetto, sia in un qualsiasi punto del documento. Il pacchetto kvoptions offre una serie di strumenti (ricalcati su quelli usati internamente in pacchetti come hyperref o geometry) per specificare le opzioni del pacchetto nella forma "chiave = valore". Verrà perciò caricato nel preambolo del nostro file di stile.

```
\ProvidesPackage{imfell}
\RequirePackage{graphics}
\RequirePackage{kvoptions}
\SetupKeyvalOptions{%
  family=IM,
  prefix=IM@,
}
```

Inoltre, se il documento viene compilato con `pdflatex`, deve essere caricata l'apposita mappa, per cui bisognerà ricorrere all'espressione condizionale `\ifpdf`.

```
\RequirePackage{ifpdf}
\ifpdf
  \pdfmapfile{+pdftex-fell.map}
\fi
```

Il punto fondamentale è che il nome delle famiglie dei font da utilizzare è costituito dal nome del font vero e proprio (o della collezione: `fell`) seguito da un numero arabo che indica il livello di legature. Il modo più semplice per specificare separatamente questi due elementi è dichiarare le variabili che le contengono, `\IM@basefamily` e `\IM@liglevel`, e utilizzare poi queste all'interno dei comandi di basso livello. Così, per passare da una famiglia all'altra e da un livello di legature all'altro si possono utilizzare i seguenti comandi:

```
\DeclareRobustCommand{\switchtoliglevel}[1]{%
  \edef\IM@tempa{#1}
  \def\IM@liglevel{\IM@tempa}%
  \fontfamily{\IM@basefamily\IM@liglevel}%
  \selectfont%
}
\DeclareRobustCommand{%
  \switchtofamily}[2][\IM@liglevel]{%
    \edef\IM@tempa{#1}
    \switchtoliglevel{#1}%
    \def\IM@basefamily{#2}%
    \fontfamily{\IM@basefamily\IM@liglevel}%
    \selectfont%
}
```

Per quanto riguarda il comando `\switchtofamily` è possibile specificare anche il livello di legature come argomento opzionale. Oltre a questi comandi generali, possono apparirne anche altri più specifici, costruiti su di essi.

```
\DeclareRobustCommand{\textpi}[2][\IM@liglevel]{%
  {\switchtofamily[#1]{pica} #2}%
}
```

Il comando `\textpi` serve a comporre il testo racchiuso nell'argomento obbligatorio, nel tipo di carattere *Pica*, specificando eventualmente anche il livello di legature desiderato.

Infine, è utile avere un comando per richiamare, in maniera semplice, un determinato ornamento; qualcosa del tipo `\fellornament{31}`, dato che è più comodo identificare un ornamento con un numero che con un nome. Per ottenere qualcosa di simile è necessario prima catalogare tutti gli ornamenti e, di conseguenza, definire un comando che faccia appunto questo. Il codice che segue è una proposta in tal senso.

```
\DeclareRobustCommand{\fellornament}[1]{%
  \csname flow@orn@#1\endcsname
}
\newcommand{\DeclareTextOrnament}[7]{%
  \expandafter \def\csname #1@orn@#2\endcsname{%
    \usefont{#3}{#4}{#5}{#6} \char#7%
    }%
}
```

A questo punto, dopo aver inserito nel file di stile la riga

```
\DeclareTextOrnament{flow}{31}{U}{flow}{m}{sc}{1}
```

il comando `\fellornament{31}` inserisce nel documento l'ornamento che si trova nella posizione "1" del font `FeFlow2`. Infatti alla forma `sc` corrisponde, secondo quanto è scritto nel file `uflow.fd`, proprio quel font (sezione 4.4).

Giunti alla conclusione di questa complessa e laboriosa installazione, i *Fell Types* sono finalmente in grado di essere utilizzati in un documento che ne sfrutti appieno le particolarità.

4.7 Un esempio

Qui di seguito (figura 4) è mostrata una pagina che riproduce l'inizio del primo capitolo dell'opera, citata nella sezione 4.2, *Pseudodoxia Epidemica, or Vulgar Errors* di Thomas Browne. Non verrà qui riprodotto il codice[29] che ne è alla base, in quanto, per la maggior parte, quel codice non ha niente a che vedere con i *Fell Types*; ma ci si limiterà ad un paio di considerazioni che sono attinenti al presente lavoro.

Innanzitutto, l'utilizzo dei tipi di carattere, in questo caso si tratta dell'intera collezione con tutte le legature attivate, è tanto semplice quanto inserire le seguenti righe:

```
\usepackage[T1]{fontenc}
\usepackage[default,liglevel=2]{imfell}
```

Il corpo del testo è composto in *Pica*, con le note a margine in *English*. Il titolo della sezione è in *Great Primer*, con l'intestazione in *Double Pica*. Il titolo del capitolo, infine è in *French Canon* e *Great Primer*. Per il capolettera è stato utilizzato il *Three Line Pica*.

29. Come al solito, però esso è consultabile nell'archivio `felltypes.tar.gz`, reperibile all'indirizzo `http://www.guit.sssup.it/download/`

THE

FIRST BOOK:

OR

GENERAL PART

CHAP. I

Of the Cauſes of Common Errors.

Introduction

THE First and Father-cauſe of common Error, is, The common infirmity of Human Nature; of whoſe deceptible condition, although perhaps there ſhould not need any other eviction, than the frequent Errors we ſhall our ſelves commit, even in the express declarement hereof: yet ſhall we illuſtrate the ſame from more infallible conſtitutions, and perſons preſumed as far from us in condition, as time, that is, our first and ingenerated forefathers. From whom as we derive our Being, and the ſeveral wounds of conſtitution; ſo, may we in ſome manner excuſe our infirmities in the depravity of thoſe parts, whoſe Traductions were pure in them, and their Originals but once removed from God. Who notwithſtanding (if posterity may take leave to judg of the fact, as they are aſſured to ſuffer in the puniſhment) were groſly deceived, in their perfection; and ſo weakly deluded in the clarity of their underſtanding, that it hath left no ſmall obſcurity in ours, How error ſhould gain upon them.

Matter of great diſpute, how our fathers could be ſo deceived

For first, They were deceived by Satan; and that not in an inviſible inſinuation, but an open and diſcoverable apparition, that is, in the form of a Serpent; whereby although there were many occaſions of ſuſpition, and ſuch as could not eaſily eſcape a weaker circumſpection, yet did the unwary apprehenſion of *Eve* take no advantage thereof. It hath therefore ſeemed ſtrange unto ſome, ſhe ſhould be deluded by a Serpent, or ſubject her reaſon to a beast, which God had ſubjected unto hers. It hath empuzzled the enquiries of others to apprehend, and enforced them unto ſtrange conceptions, to make out, how without fear or doubt ſhe could diſcourſe with ſuch a creature, or hear a Serpent ſpeak, without ſuſpition of Impoſture. The wits of others have been ſo bold, as to accuſe her ſimplicity, in receiving his Temptation ſo coldly; and when ſuch ſpecious effects of the fruit were Promiſed, as to make them like God; not to deſire, at least not to wonder he purſued not that benefit himſelf. And had it been their own caſe, would perhaps have replied. If the tast of this Fruit maketh the eaters like *Gods*, why remainest

FIGURA 4: Capitolo iniziale di *Pseudodoxia Epidemica, or Vulgar Errors*, composto con i *FellTypes*

I disegni ornamentali sono stati ottenuti componendo insieme uno o più ornamenti provenienti dai due font *Flowers*. Ciò può essere fatto utilizzando, a basso livello, la primitiva di TEX `\xleaders`, che consente di ripetere un carattere (o meglio: una scatola che può contenere anche più di un carattere) fino a riempire completamente la dimensione specificata. Il problema è che, se tale dimensione non è coperta da un numero intero di elementi, `\xleaders` lascia ovviamente degli spazi bianchi di uguali dimensioni tra i vari blocchi. Questo problema è stato risolto, dato che l'elemento ornamentale può essere considerato un elemento grafico non assoggettabile ai vincoli della griglia tipografica, disponendo i suddetti blocchi su una dimensione il più possibile vicina a quella voluta e riscalando poi il risultato. In questo modo, è facilissimo inserire i due elementi in questione inserendo rispettivamente le due righe di codice che seguono.

```
\OrnamentRule{11}{2}{3}{12}{\textwidth}
\OrnamentRule{51}{51}{51}{51}{.5\textwidth}
```

Il codice per `\OrnamentRule` è incluso nel file `imfell.sty` e può qui essere consultato.

5 Conclusioni

5.1 Problemi residui

La presenza (e l'uso) di caratteri non standard nei *Fell Types* rende problematica l'estrazione di testo da un PDF che includa tali font. Infatti l'applicazione che visualizza il documento deve essere istruita ad interpretare tali caratteri come una sequenza di uno o più caratteri standard (ad esempio: `ct` → `c t`). Questo è possibile se l'applicazione trova, all'interno del documento, una tabella che, per ogni carattere non standard, associa all'indice del carattere il valore Unicode dei suoi componenti. Purtroppo solo il percorso di compilazione `tex` → `dvips` → `ps2pdf`, è in grado di inserire nel PDF una mappa /ToUnicode, e anche in questo caso limitatamente a caratteri presenti nello standard Unicode[30] (ad esempio `longs`, `longst`, `ct`, ma non `longsh` e `ll`).

`pdftex` ha un paio di primitive non documentate, `\pdfgentounicode` e `\pdfglyphtounicode`, che permettono di generare la mappa /ToUnicode e di personalizzarla, aggiungendo specifiche per caratteri non contenuti nello standard Unicode. Sfortunatamente queste primitive funzionano solo con font TYPE1.

5.2 Risultati

Il percorso affrontato in questo lavoro ha mostrato come è possibile, con TEX, sfruttare, in una maniera semplice per l'utente finale, tutte le caratteristiche di un font.

Partendo da una collezione di font ricca di caratteri inconsueti (i quali, insieme al *kerning* accurato, rappresentano la caratteristica saliente dei *Fell Types*) e difficilmente accessibili dall'interno di un programma di videoscrittura, in assenza di tabelle di sostituzione come quelle OPENTYPE, siamo stati in grado di offrire all'utente finale un'interfaccia minimale, non intrusiva, e tuttavia capace di fornirgli tutti gli strumenti necessari per gestire la ricchezza e la complessità di questi font.

Certamente tutto questo viene poi scontato con il dover affrontare un processo di installazione che non può assolutamente essere definito semplice, né può essere riportato a procedure standard da applicare ciecamente, almeno in casi complessi come quello appena visto. Tuttavia anche questa considerazione negativa è attenuata dal fatto che, grazie al carattere aperto di TEX, una volta prodotta una soluzione, questa viene di solito messa a disposizione della comunità che può utilizzarla, studiarla e migliorarla, soprattutto, senza che ogni utente debba ripercorrere ogni volta la stessa strada.

30. Le legature, e altre "forme di presentazione" non sono considerate caratteri; quelle presenti nello standard unicode lo sono solo per motivi di compatibilità. Si veda a questo proposito `http://www.unicode.org/faq/ligature_digraph.html#7`

Riferimenti bibliografici

LATEX3 PROJECT TEAM (2005). «LATEX 2ε font selection». `http://www.ctan.org/tex-archive/macros/latex/doc/fntguide.pdf`.

BECCARI, C. (2007). «Guida all'arte della composizione tipografica con LATEX». `http://www.guit.sssup.it/downloads/GuidaGuIT.pdf`.

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

— (1990). «Virtual fonts: More fun for grand wizards». *TUGboat*, **11** (1), pp. 13–23.

LEHMANN, P. (2004). «The Font Installation Guide». `http://www.ctan.org/tex-archive/info/Type1fonts/fontinstallationguide/fontinstallationguide.pdf`.

MITTELBACH, F., GOOSSENS, M. *et al.* (2004). *The LATEX companion*. Addison Wesley, 2ª edizione.

ZANNARINI, E. e VAVASSORI, E. G. (2005). «LATEX e i font: installazione pratica». In *Secondo convegno nazionale su TEX, LATEX e tipografia digitale*. Pisa, pp. 107–122.

▷ Massimiliano Dominici
`mlgdominici@interfree.it`