LaTeX–PDF–Howto

Marco Pratesi Marco Latini Michele Antonecchia

Versione 1.1.1 - 22 febbraio 2003

Sommario

Il presente Howto tratta la preparazione di documenti in formato PDF (Portable Document Format) con LaTeX. Vengono presentati più metodi e possibili approcci, cercando di evidenziare vantaggi e svantaggi di ciascuno di essi in situazioni diverse. Viene discussa anche la preparazione di slide PDF con LaTeX e vengono forniti dei semplici esempi, orientati soprattutto alla presentazione di contenuti scientifici.

Si suppone che il lettore conosca già il LaTeX: la trattazione dell'uso di LaTeX sarebbe senz'altro al di là degli scopi del presente Howto ed è già validamente affrontata in numerosi testi, molti dei quali sono anche liberamente reperibili sulla rete.

Indice

1 Disclaimer

No liability for the contents of this document can be accepted. Use the concepts, examples, and other content at your own risk.

All copyrights are held by their respective owners, unless specifically noted otherwise. Use of a term in this document should not be regarded as affecting the validity of any trademark or service mark.

Naming of particular products or brands should not be seen as endorsements, with the exception of the term "GNU/Linux". We wholeheartedly endorse the use of GNU/Linux in every situation where it is appropriate. It is an extremely versatile, stable, and robust operating system.

2 Introduzione

L'utilizzo del sistema di formattazione testi TeX in ambienti Unix è un requisito fondamentale per la creazione di documenti stilisticamente corretti e dal grande impatto visivo. Al fine di ottenere un output in formato PDF, e quindi portabile senza troppe difficoltà praticamente su qualsiasi piattaforma, sono stati introdotti alcuni pacchetti in grado di svolgere al meglio questa funzione. Esistono, e sono stati più volte rilevati, problemi relativi al risultato della conversione che alcune volte non corrisponde affatto al risultato che si attendeva, pur essendo la sintassi ed i comandi TeX perfettamente corretti. Questo Howto è nato ed è stato sviluppato per far fronte proprio a questi problemi.
Verranno trattati in particolare i seguenti pacchetti presenti ormai su tutte le piattaforme Linux:

- `ps2pdf`
- `pdflatex`
- `dvipdfm`

Riguardo a `dvipdfm`, si precisa che fa parte dei pacchetti di base di Debian GNU/Linux e Mandrake Linux e può essere reperito nella sezione `contrib` di Red Hat Linux.

2.1 Commenti

Ogni suggerimento, codice, pacchetto riportato su questo Howto è stato preventivamente provato sulle nostre macchine in misura ragionevole. Tuttavia non possiamo escludere un mancato o non corretto funzionamento di quanto riportato nel seguito (il che potrebbe anche dipendere dalle configurazioni usate dall'utente finale). Qualora rilevaste qualche errore o mancanza all'interno di questo Howto **siete pregati di comunicarcelo**. Provvederemo quindi alla sua correzione al più presto. Probabilmente non potremo rispondere alle e–mail di tutti, quindi cercate di comprendere se non ricevete risposta immediata e portate pazienza.

2.2 Perché è stato scritto questo Howto

Come membri del LUG Roma, iscritti pertanto alla sua mailing-list, abbiamo notato che anche utenti esperti hanno trovato difficoltà nel reperire informazioni su come usare LaTeX

per preparare documenti in formato PDF anziché PS come si fa classicamente. Alla luce di questa considerazione si è deciso di convogliare le esperienze in un unico documento che possa aiutare in questo compito. Nel seguito, si assume che il lettore sia già a conoscenza di LATEX e del suo uso per la produzione di documenti in formato DVI (Device Independent) e PS (PostScript). Comunque, abbiamo inserito tra i riferimenti bibliografici due manuali LATEX che abbiamo trovato utili, interessanti, non eccessivamente estesi e disponibili anche in italiano [1], [2].

2.3 Documentazione su TEX e LATEX

È possibile trovarne in notevole quantità e fare il punto in merito è senz'altro al di là degli scopi del presente documento; comunque, come già detto, in bibliografia abbiamo citato due manuali che ci sono sembrati interessanti e che utilizziamo attualmente [1], [2].

3 Ripartiamo dal classico PostScript: ps2pdf

Se si sa già come produrre un documento PS tramite LATEX e non si ha a disposizione più di 5 minuti per produrre un documento PDF (compreso il tempo a disposizione per capire come fare :-) , sicuramente la via più rapida consiste nell'uso di `ps2pdf`, uno script di shell `sh` che fa uso di GhostScript, in particolare della sua device `pdfwrite`, per convertire un PostScript in un corrispondente PDF. Le versioni più recenti di GhostScript includono più di uno script per la conversione da ps a pdf: `ps2pdf`, `ps2pdf12`, `ps2pdf13`, `ps2pdf14`, da usare a seconda della versione di formato pdf che si vuole ottenere (1.2, 1.3, 1.4); tali script fanno riferimento allo script `ps2pdfwr`.
L'uso di `ps2pdf` è decisamente immediato:

```
ps2pdf [nomefile].ps
```

Per una spiegazione più completa sul suo uso, si rimanda alla sua pagina di manuale.
Il cuore di `ps2pdfwr` corrisponde alla sua ultima riga, in cui si invoca `gs`, cioè il GhostScript:

```
exec gs $OPTIONS -q -dNOPAUSE -dBATCH -sDEVICE=pdfwrite
        -sOutputFile="$outfile" $OPTIONS -c .setpdfwrite -f "$infile"
```

Se il `gs` è stato compilato in modo da produrre, per default, una pagina di formato letter e invece si desidera produrre una pagina di formato A4, si può modificare lo script (o una sua copia di utente) aggiungendo alla chiamata di `gs` l'opzione `-sPAPERSIZE=a4` e quindi sostituendo la riga riportata sopra con la seguente:

```
exec gs $OPTIONS -q -dNOPAUSE -dBATCH -sDEVICE=pdfwrite -sPAPERSIZE=a4
        -sOutputFile="$outfile" $OPTIONS -c .setpdfwrite -f "$infile"
```

Le versioni meno recenti di GhostScript includono un solo script per la conversione da ps a pdf: `ps2pdf`; in tal caso, se necessario, una modifica del tutto analoga a quella appena indicata va fatta a quest'ultimo script.

Le versioni recenti di GhostScript, oltre agli script `ps2pdf`, includono anche uno script `dvipdf` (da non confondere con il `dvipdfm`, trattato più avanti). `dvipdf` non fa altro che passare "al volo" dal DVI al PDF usando `dvips` e `gs` con la device `pdfwrite`, quindi in

pratica il risultato è lo stesso che si ottiene passando da DVI a PS mediante `dvips` e da PS a PDF mediante `ps2pdf`; tuttavia, a meno di particolari motivi, sembra preferibile usare `dvips` e `ps2pdf` anziché direttamente `dvipdf`, perché così si può scegliere quale script usare tra `ps2pdf`, `ps2pdf12`, `ps2pdf13` e `ps2pdf14` e, quindi, decidere la versione di formato pdf che si vuole ottenere.

Ora facciamo qualche considerazione sui documenti PDF ottenuti come sopra a partire dal PS mediante `ps2pdf`. Di default, `dvips`, nel produrre il PS a partire dal DVI, produce un documento contenente dei font "Type 3", "bitmapped", non scalabili. A partire da un documento PS ottenuto mediante `dvips`, in generale il GhostScript non è in grado di ottenere un documento PDF contenente font scalabili, cioè vettoriali anziché bitmap. L'effetto più evidente sta nel fatto che, se si visualizza il documento PDF usando Acrobat Reader, la lettura risulta molto faticosa, le lettere appaiono sgranate, sfocate; questo è dovuto al fatto che un "matching" tra la risoluzione dello schermo, lo zoom della visualizzazione e la risoluzione dei font utilizzati è estremamente improbabile, impossibile in termini pratici. Lo stesso documento viene visualizzato bene se si usa GhostView utilizzando l'antialiasing del GhostScript; ma, per ciò che riguarda l'aumento della risoluzione scelta oltre certi limiti non migliora il risultato, dato che i font bitmap sono di fatti a risoluzione fissa.
Per superare questo problema è necessario fare in modo che `dvips` usi dei font "Type 1", scalabili; a tale scopo, l'utente deve creare nella sua home directory un file

```
~/.dvipsrc
```

Contenente almeno le seguenti righe:

```
p +psfonts.cmz
p +psfonts.amz
```

(gli utenti Miktex devono aggiungere le stesse righe al file `\texmf\dvips\config\config.ps`)
Il problema può essere risolto anche scegliendo di non usare i Computer Modern Font (CMF), normalmente usati con LaTeX, così belli da essere un vanto oltre che un segno distintivo dei documenti prodotti con LaTeX. Usando i font Times invece dei Computer Modern Font, si producono documenti PS con font scalabili anche senza usare l'accorgimento presentato sopra, cosicché si può ottenere un documento PDF convertendo il corrispondente PS con `psp2df`, senza incontrare i problemi appena citati. Per fare questo è necessario includere il package `times` nel preambolo del `.tex`:

```
\usepackage{times}
```

Bisogna tenere presente che tale package comporta l'uso dei font Times solo nel testo. Se si vuole usare i font Times anche per le equazioni e il modo matematico in generale, è necessario includere anche il package `mathptm`:

```
\usepackage{mathptm}
```

Tuttavia, l'uso di questi package a nostro avviso riduce la bellezza e la qualità delle metriche tipiche dei documenti prodotti con LaTeX.

Ulteriori informazioni riguardanti l'uso dei font nei documenti PDF prodotti con LaTeX sono reperibili in [3], nella sezione relativa alla produzione di documenti PDF.

4 Perché non usare ps2pdf

Il formato PDF offre una potenza espressiva superiore a quella offerta dal PS, dato che, ad esempio, permette di utilizzare l'ipertesto, i bookmarks e i thumbnails, a cui si fa riferimento nel seguito. Se il PDF viene prodotto mediante conversione a partire dal PS, si ottiene un documento che non può avere più potenza espressiva di quella offerta dal formato PostScript, dato che una conversione non può aumentare il contenuto informativo di partenza.
Quindi, se possibile, è meglio procedere come spiegato nel seguito del presente Howto.

5 Come fare senza ps2pdf: pdflatex, dvipdfm

Se il documento LaTeX in questione contiene solo testo e, soprattutto, non include figure e immagini, lo stesso sorgente `.tex` che si compila con il comando `latex` può essere compilato con il comando `pdflatex`, producendo così un documento PDF anziché PS. In alternativa, si può usare come al solito il comando `latex`, che permette di ottenere il documento DVI; quindi, anziché usare `dvips` per produrre il documento PS a partire dal DVI, si può usare il comando `dvipdfm` per produrre il documento PDF a partire dal DVI.

Ovviamente, la questione è più complessa se nel documento si devono includere figure, dato che la compilazione `latex` si aspetta di includere figure in formato PS o EPS, mentre la compilazione `pdflatex` non prevede l'inclusione di figure in tali formati e normalmente fa riferimento all'inclusione di figure in altri formati, tra cui PDF e PNG. Riteniamo molto interessante il poter preparare un documento `.tex` che permetta di produrre con compilazioni diverse sia il formato PS che il formato PDF, a patto di avere a disposizione le figure da includere almeno in due formati: uno adatto a `latex` (PS o EPS), un altro adatto a `pdflatex` (ad esempio, PDF o PNG).

Nella parte finale del documento è affrontata la preparazione/conversione di figure e immagini nei formati menzionati.

5.1 Un preambolo "universale"

Quello che segue è il preambolo utilizzato per il presente Howto ed è molto comodo perché può essere usato **senza necessità di nessuna modifica** in tutti i seguenti casi

- produzione del DVI con il comando `latex` e quindi del PS con il comando `dvips` (uso "classico" di LaTeX)
- produzione del PDF con il comando `pdflatex`
- produzione del DVI con il comando `latex` e quindi del PDF con il comando `dvipdfm`

tutto questo anche gestendo parametricamente l'inclusione delle figure (che viene affrontata in dettaglio più avanti) e permettendo di gestire l'ipertesto e i thumbnails delle pagine se si produce un documento PDF.

```
01 \ifx\pdfoutput\undefined % We are not running pdftex
02 \documentclass[12pt,a4paper]{article}
03 \usepackage[dvips]{graphicx}
```

```
04 %\usepackage{psfig}
05 %\usepackage{epsfig}
06 \RequirePackage[dvipdfm,hyperindex]{hyperref}
07 %\RequirePackage[dvipdfm,colorlinks,hyperindex]{hyperref}
08 \else
09 \documentclass[pdftex,12pt,a4paper]{article}
10 \usepackage[pdftex]{graphicx}
11 \DeclareGraphicsExtensions{.pdf,.png,.jpg,.mps}
12 \RequirePackage[hyperindex]{hyperref}
13 %\RequirePackage[colorlinks,hyperindex]{hyperref}
14 \usepackage{thumbpdf}
15 \fi
```

La parte da riga 02 a riga 07 viene considerata se si usa il comando `latex`, più precisamente, se non si sta producendo un output in formato PDF, come si può intuire dall'`if` alla riga 01. Se invece si sta producendo un output in formato PDF, viene considerata la parte da riga 09 a riga 14.

Ricordiamo che a partire dal carattere % fino alla fine della linea viene ignorato tutto, dato che % viene usato in LaTeX come carattere per iniziare una parte "commentata", che il compilatore deve ignorare. Quindi, nell'esempio sopra, le righe 04, 05, 07 e 14 vengono ignorate in ogni caso. Sono state inserite perché costituiscono delle interessanti alternative alle corrispondenti linee non "commentate".

Le righe 02 e 09 dovrebbero avere un significato già noto agli utenti LaTeX e fissano delle impostazioni generali per l'intero documento; si rimarca la presenza di `pdftex` tra le opzioni usate per la `documentclass` nel caso in cui l'output sia in PDF (questa è l'unica differenza tra le righe 02 e 09).

Anche le righe 03 e 10 sono praticamente identiche, anche se il package `graphicx` viene caricato con un driver diverso a seconda dei casi: `dvips` se non si sta uscendo in PDF (se si sta uscendo in DVI), `pdftex` se si sta uscendo in PDF. Le righe 04 e 05 costituiscono un'alternativa alla riga 03: nel caso in cui si vuole uscire in DVI prima e PS poi, anziché il package `graphicx` si può usare `psfig` oppure `epsfig`, package che permettono di includere figure PS ed EPS, ma che nel seguito non verranno presi in considerazione.

Nella riga 11 sono dichiarati i formati grafici utilizzabili con il package `graphicx` quando viene usato con il driver `pdftex`, cioè `.pdf,.png,.jpg,.mps` (`.mps` corrisponde a metapost output).

La riga 12 carica il package `hyperref`, che serve a ottenere, in corrispondenza dei riferimenti incrociati (ad es. riferimenti a voci bibliografiche, a figure, a tabelle, ecc.), dei link ipertestuali analoghi a quelli usati sulle pagine web (si precisa che l'ipertesto può essere presente nei formati DVI e PDF, ma viene perso se si passa al formato PS, che non contempla tali costrutti). Viene caricato il driver `hyperindex` per produrre un documento che contenga l'indice nella sezione dei Bookmarks (il che ne facilita molto la "navigazione"). Usando la riga 12 si fa in modo che i link siano costituiti da scritte nere contornate a colori (colori che non verranno stampati e che saranno diversi a seconda del tipo di riferimento incrociato). La riga 13 può essere usata in alternativa alla 12 per ottenere dei link colorati senza contorni, cioè per fare in modo che le scritte con link ipertestuali siano a colori anziché nere, ma senza contorni.

Le righe 06 e 07 sono del tutto analoghe alle righe 12 e 13: l'unica differenza sta nel fatto che

viene caricato anche il driver `dvipdfm`, per poter ottenere lo stesso tipo di indice nel caso in cui il PDF venga prodotto utilizzando `dvipdfm`, cioè passando per il DVI. Si tenga presente che caricando tale driver si perde la possibilità di utilizzare l'ipertesto nel DVI; quindi, se non si ha intenzione di usare `dvipdfm` e si pensa di usare solo `latex` e `pdflatex`, può essere opportuno non caricarlo.

La riga 14 serve a includere un package utile per produrre tramite `pdflatex` dei PDF contenenti anche le miniature (thumbnails) di ogni singola pagina. Tale aspetto è affrontato più avanti in una apposita sezione.

Infine, può risultare utile la parte che abbiamo usato subito dopo, riportata di seguito:

```
\hypersetup{
pdftitle={LaTeX-PDF-Howto},
pdfsubject={Come produrre documenti PDF con LaTeX},
pdfauthor={Marco Pratesi, Marco Latini, Michele Antonecchia},
pdfkeywords={LaTeX, PDF, Howto, pdflatex, dvipdfm},
%pdfpagemode=FullScreen
}
```

che serve ad inserire informazioni generali sul documento, visualizzabili ad esempio con Acrobat Reader. In particolare, `pdfpagemode=FullScreen` serve a ottenere un documento che mandi Acrobat Reader direttamente in "Full Screen Mode" (cioè a pieno schermo) quando viene caricato. In questo caso la corrispondente riga è commentata perché sul presente documento tale effetto sarebbe inopportuno; tuttavia, nel caso in cui si stiano preparando delle slide (tale caso è affrontato più avanti in questo Howto), tale effetto può essere voluto e quindi può essere opportuno inserire la riga in questione senza commentarla.

5.2 Inclusione "parametrica" di una figura

Partiamo subito con un esempio che spiega più di quanto possano molte parole... Il seguente codice permette di includere la Fig. 1:

```
01 \begin{figure}
02 \centerline{\includegraphics[width=0.7\textwidth]{category}}
03 \caption{Categories of Free and Non-Free Software, taken from
04 {\tt http://www.gnu.org/philosophy/categories.html}}
05 \label{software-categories}
06 \end{figure}
```

Nella riga 02 non viene specificata l'estensione del file da includere, viene indicato solo il nome file `category`. Sono disponibili sia `category.eps` che `category.pdf`, il primo ottenuto da `category.fig`, prodotto con XFig, il secondo ottenuto dal primo mediante `epstopdf` (XFig ed `epstopdf` sono menzionati più avanti, nella sezione dedicata ai programmi utilizzati per produrre le figure). Se si compila con il comando `latex`, viene incluso il file `category.eps`; se invece si compila con il comando `pdflatex`, viene incluso il file `category.pdf`; in entrambi i casi si ottiene l'inclusione della figura nel formato adatto.

Altro esempio; il seguente codice permette di includere la Fig. 2

```
01 \begin{figure}
```

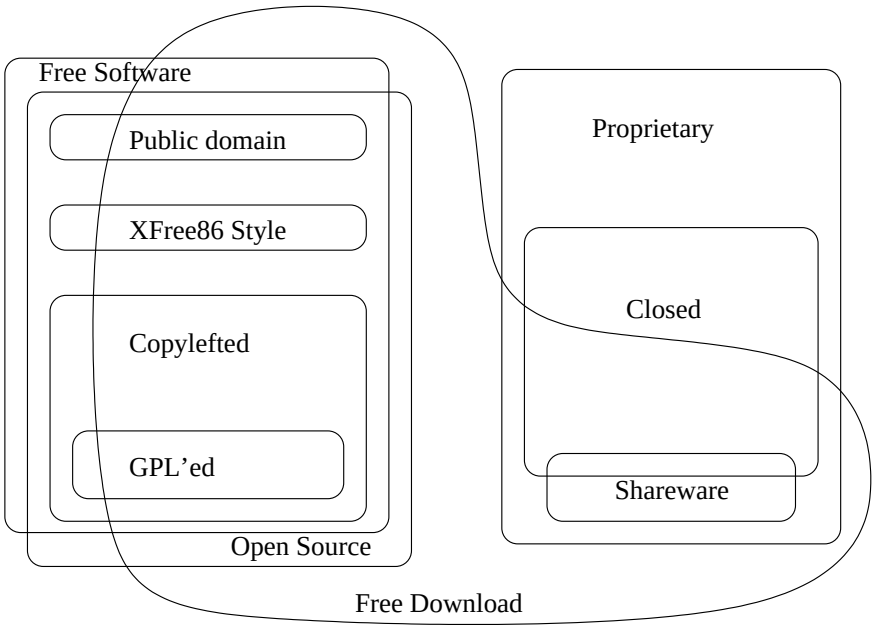

Figura 1: Categories of Free and Non-Free Software, taken from `http://www.gnu.org/philosophy/categories.html`

```
02 \centerline{\includegraphics[width=0.7\textwidth]{linus3}}
03 \caption{Linus Benedict Torvalds, the ‘‘father’’ of Linux}
05 \label{Linus}
06 \end{figure}
```

Sono disponibili le figure `linus3.eps` e `linus3.png`, entrambe ottenute mediante The GIMP (menzionato più avanti) convertendo nei formati desiderati una famosa immagine di Linus Torvalds, cioè `linus3.gif`. La prima, `linus3.eps`, viene inclusa se si usa il comando `latex`; la seconda, `linus3.png`, viene inclusa se si usa il comando `pdflatex`.

5.3 Uso di `pdflatex`

Seguiti gli accorgimenti appena illustrati, produrre un PDF con `pdflatex` è davvero banale: basta usare il comando `pdflatex` invece del classico `latex`:

```
pdflatex [nomefile].tex
```

5.4 Uso di `dvipdfm`

Semplice come bere un bicchier d'acqua: è sufficiente usare il comando

```
dvipdfm [nomefile].dvi
```

dopo aver effettuato la "classica" compilazione con il comando `latex`. Le figure PS/EPS incluse nel DVI vengono convertite "on the fly" in fase di compilazione utilizzando GhostScript. Si precisa che, se si intende produrre il PDF con `dvipdfm` seguendo l'approccio mostrato fin qui, non è necessario né utilizzare un preambolo parametrico come quello mostrato né includere parametricamente le figure, è sufficiente usare `dvipdfm` laddove classicamente viene

Figura 2: Linus Benedict Torvalds, the "father" of Linux

usato `dvips`. Ovviamente, se si desidera anche gestire l'ipertesto, l'indice nei Bookmarks e i Thumbnails (caratteristiche che il formato PDF offre in più rispetto al PS), è necessario usare un preambolo più ricco, come già illustrato, e procedere come è indicato più avanti nella sezione dedicata all'inserimento dei thumbnails.

5.5 Un confronto tra `pdflatex` e `dvipdfm`

La prima differenza che salta all'occhio confrontando `pdflatex` e `dvipdfm` sta nella differenza tra i due approcci. Usando `pdflatex` si passa direttamente dal `.tex` al `.pdf`; invece, se si usa `dvipdfm`, si procede in maniera del tutto analoga all'uso "classico" di LaTeX: prima di tutto si produce il DVI compilando con il comando `latex`, quindi, a partire dal DVI e dalle figure incluse, si ottiene il PDF (PS) con il comando `dvipdfm` (`dvips`). Ne consegue che, usando `dvipdfm`,

- è necessario un passaggio in più rispetto all'uso di `pdflatex`
- si producono documenti PS e PDF a partire dallo stesso DVI

Dall'ultimo punto consegue che, usando `dvipdfm`, è impossibile ottenere uno stesso documento con impaginazioni diverse nei formati PS e PDF, dato che entrambi scaturiscono dallo stesso DVI e, nel formato DVI, l'impaginazione è già fissata. Questo invece risulta possibile (anche se davvero molto remoto) se il PS viene prodotto a partire dal DVI e il PDF viene prodotto a parte mediante `pdflatex`. D'altra parte si può osservare anche che

- il PS e il PDF hanno la stessa impaginazione anche usando `pdflatex`, se il PS viene ottenuto a partire dal PDF (ci sono molti strumenti per farlo) anziché a partire dal DVI mediante `dvips`
- si tratta di considerazioni non di interesse se si deve produrre il documento solo in formato PDF

Un'altra differenza molto importante sta nel fatto che, in fase di compilazione, `dvipdfm` converte "on the fly" tramite GhostScript tutte le figure incluse nel documento; questo, se da una parte facilita la gestione delle figure, sollevando l'utente da tutte le problematiche connesse, significa che tale conversione avviene ad ogni compilazione, anziché una volta per tutte; quindi le compilazioni richiedono più tempo di quello necessario se si usa `pdflatex` e questo può risultare pesante se le figure da includere sono molte (anche per questo motivo, di seguito si presenta un approccio leggermente diverso per l'inclusione delle figure quando si usa `dvipdfm`).
La contropartita, con `pdflatex`, sta nel fatto che, se si producono figure PS/EPS che poi vanno convertite in PDF per essere incluse nel documento, si può avere spesso il dubbio

> "*ho convertito tutte quelle che ho aggiornato? Non ne avrò dimenticata qualcuna? Forse, per sicurezza, mi conviene riconvertirle tutte... ma mi sa che nel riconvertirle tutte qualche comando errato mi scappa... chissà come verrà fuori questo documento, spero di non fare stupidaggini...*"

In questi casi, la prima cosa che si tende a fare è preparare un semplice script con tutti i comandi di conversione, ad esempio

```
epstopdf figura1.eps
epstopdf figura2.eps
epstopdf figura3.eps
epstopdf figura4.eps
epstopdf figura5.eps
epstopdf figura6.eps
```

Tale semplice script ci solleva dal timore di dare comandi errati o dimenticare qualche conversione, ma ci costringe comunque a riconvertire anche le figure che non sono state modificate nel frattempo.
Se si vuole riconvertire solo quelle non ancora convertite o aggiornate nel frattempo, è sufficiente ricordarsi dell'esistenza di uno strumento potente come il `make` e utilizzare un `Makefile` siffatto (leggere la documentazione di `make` per saperne di più):

```
Makefile
-------------------------------------------------

all:    figura1.pdf figura2.pdf figura3.pdf \
        figura4.pdf figura5.pdf figura6.pdf

%.pdf: %.eps
        epstopdf $<

clean:
        rm -rf figura?.pdf

-------------------------------------------------
```

Una volta prodotto un `Makefile` di questo tipo, basta dare il comando `make` per convertire solo quelle che non sono state ancora convertite e quelle i cui `.eps` sono stati aggiornati nel

frattempo. Usando il comando `make clean` si cancellano tutte le figure `.pdf` lasciando solo l'indispensabile, cioè solo le figure `.eps` originali.
Si ritiene che il `Makefile` appena presentato sia sufficientemente autoesplicativo; tuttavia, per renderlo più chiaro, se ne propone di seguito una versione equivalente più estesa e ridondante

`Makefile`

```
-------------------------------------------------

all:    figura1.pdf figura2.pdf figura3.pdf \
        figura4.pdf figura5.pdf figura6.pdf

figura1.pdf: figura1.eps
        epstopdf figura1.eps

figura2.pdf: figura2.eps
        epstopdf figura2.eps

figura3.pdf: figura3.eps
        epstopdf figura3.eps

figura4.pdf: figura4.eps
        epstopdf figura4.eps

figura5.pdf: figura5.eps
        epstopdf figura5.eps

figura6.pdf: figura6.eps
        epstopdf figura6.eps

clean:
        rm -rf figura1.pdf figura2.pdf figura3.pdf \
               figura4.pdf figura5.pdf figura6.pdf

-------------------------------------------------
```

Si fa presente che in un `Makefile` non si devono usare 8 spazi laddove va usato un `TAB`; si rimanda alla documentazione di `make` per informazioni più dettagliate in merito.

Una breve nota sulla versione di GhostScript utilizzata per convertire i PS/EPS in PDF: se chi stampa il documento usa un sistema di stampa basato su GhostScript (come normalmente accade su Linux) ma con una versione più vecchia di quella utilizzata per la conversione, si possono avere problemi nella stampa. Per minimizzare tali problemi, ci si dovrebbe chiedere quale insieme di piattaforme si vuole supportare senza rischi di questo tipo e quindi usare, per le conversioni da PS a PDF, la versione di GhostScript più vecchia tra quelle che si desidera supportare. Questo non significa che se sul sistema è installata una versione più nuova di GhostScript si deve effettuare un downgrade di GhostScript, dato che è sufficiente fare un'installazione da utente di una versione di GhostScript "sufficientemente vecchia", ad esempio copiando in una directory di utente gli eseguibili presi da un pacchetto vecchio della

distribuzione GNU/Linux che si sta usando. Ad esempio, fino alla metà del 2002, tra le ultime release stabili delle distribuzioni Linux, la Debian 2.2 era quella che includeva la versione più vecchia di GhostScript, cioè la 5.10. Chi ad esempio usava Red Hat 6.2 o una versione successiva, aveva a disposizione GhostScript 5.50 o una versione successiva; per non causare problemi agli utenti Debian, si poteva installare in una apposita directory il GhostScript 5.10 della Red Hat 6.1, quindi copiare in tale directory lo script `epstopdf` e modificarlo indicando il path completo di "`gs`" 5.10, evitando che per la conversione venga eseguito il 5.50 disponibile nella distribuzione usata. Ovviamente, nell'effettuare tale conversione, nel comando bisognava indicare il percorso completo della versione così modificata dello script `epstopdf`.
Analoghe considerazioni possono essere fatte riguardo allo script `ps2pdf`, considerato all'inizio del presente documento.

5.6 Un approccio diverso per l'inclusione delle figure

Consideriamo ora un approccio leggermente diverso (e più comodo per l'uso di `dvipdfm`) per ciò che riguarda l'inclusione delle figure nel documento. Supponiamo ad esempio di non essere interessati a poter ottenere il documento anche in formato PostScript e/o di non desiderare che `dvipdfm` converta tutte le immagini da ps/eps ogni volta che si produce il documento PDF a partire dal documento DVI. Ebbene, `dvipdfm` permette di ottenere il PDF a partire dal DVI anche includendo immagini in formati diversi da ps/eps. Ad esempio, può usare immagini che siano già in formato pdf e/o immagini nei formati png e jpeg. Supponiamo ad esempio di voler usare solo la versione png dell'immagine di Linus Torvalds; in tal caso l'istruzione di inclusione di tale immagine può essere scritta in maniera meno generale, aggiungendo anche l'estensione `.png` al nome del file, dato che `dvipdfm` è in grado di includere direttamente immagini in formato png (e tale scrittura "meno generale" non inficia minimamente la compilabilità del documento con `pdflatex`). Si pone però un problema: `dvipdfm` serve a ottenere il documento PDF a partire dal documento DVI; quindi, prima di ottenere il PDF, è necessario ottenere il DVI tramite il comando `latex`; ma con il preambolo proposto finora, se si prova a includere l'immagine .png in questione, LaTeX risponde con il seguente messaggio di errore:

```
! LaTeX Error: Cannot determine size of graphic in linus3.png (no BoundingBox).
```

dicendoci, in pratica, che non è in grado di capire quanto sia grande l'immagine png in questione e quali siano le coordinate dei suoi confini; questo problema non si pone con una normale immagine ps/eps, ma con immagini png, jpeg, pdf, si pone senz'altro. A questo punto, ci viene in aiuto il comando `ebb`, che fa parte del pacchetto `dvipdfm`, che, come dice la sua pagina di manuale, legge immagini nei formati jpeg, png o pdf e produce i corrispondenti "bounding box files", che sono file con lo stesso nome file ed estensione `.bb`, che contengono le informazioni di bounding box dell'immagine, cioè quelle che servono a `latex` per produrre un DVI che includa ad esempio l'immagine png di cui stiamo discutendo. A questo punto, possiamo lanciare il comando

```
ebb linus3.png
```

e poi fare lo stesso per le altre immagini incluse nel documento, dopodiché, nel "preambolo universale" usato finora, bisogna sostituire la riga

```
\documentclass[12pt,a4paper]{article}
```

con la riga

```
\documentclass[dvipdfm,12pt,a4paper]{article}
```

per poi procedere come già visto per creare prima il documento DVI e poi il documento PDF finale. C'è da notare una differenza rispetto all'uso di `dvipdfm` visto prima: in questo caso, il previewer `xdvi` non è in grado di mostrarci una preview dell'immagine inclusa, dato che tale immagine non è in formato ps/eps; al suo posto si vede uno spazio vuoto, che, nel PDF finale, sarà occupato dall'immagine in questione. C'è però un interessante vantaggio nell'usare questo approccio: `dvipdfm` non deve convertire al volo le immagini dal formato ps/eps ogni volta che si deve ottenere il PDF a partire dal DVI; ad esempio, utilizza ogni volta direttamente `category.pdf` senza bisogno di convertire `category.eps` ad ogni aggiornamento del documento. Inoltre, non è più necessario produrre delle immagini ps/eps se non si desidera utilizzarle, e questo può essere un notevole sollievo quando è necessario includere nel documento delle immagini non vettoriali, ad esempio ottenute scannerizzando delle foto, che in formato ps/eps potrebbero occupare molto spazio su disco e richiedere dei tempi di conversione non trascurabili a ogni esecuzione di `dvipdfm`.

6 Un suggerimento banale ma piuttosto utile...

Per visualizzare la versione PDF di un documento in fase di stesura, è preferibile usare `xpdf` oppure `gv` piuttosto che Acrobat Reader, perché questi programmi, anche se mancano di alcune funzionalità offerte da Acrobat Reader, risultano più comodi per la preview di un documento il cui contenuto cambia durante la sessione di lavoro. Infatti, per ricaricare un documento già aperto, su `xpdf` è sufficiente premere "`r`" e su `gv` basta selezionare "File" e "Reopen", oppure si può attivare l'opzione "Watch File" tra le opzioni di ghostscript perché il documento sia controllato periodicamente e ricaricato ogni volta che risulta modificato. Inoltre, `xpdf` e `gv` sono dei software liberi (con licenza GNU GPL), al contrario di Acrobat Reader, che è un software gratuito ma proprietario e a sorgenti chiusi.

7 Come inserire i thumbnails...

La possibilità di introdurre Thumbnails nel documento PDF consente di accentuare il carattere professionale dello stesso.

7.1 ... usando `pdflatex`

Includere il package `thumbpdf` nel preambolo è necessario ma non sufficiente. D'altronde lo si scoprirebbe dando un'occhiata al file `.log` prodotto dalla compilazione. Occorre infatti creare questi thumbnails attraverso il comando `thumbpdf` come segue.

Nella directory dove è presente il nostro file digitiamo:

```
thumbpdf [nomefile.pdf]
```

In questo modo verranno creati tanti file .png quante sono le pagine del nostro documento e rappresentanti ciascuno ognuna di queste. Verranno inoltre creati dei file `thumbta` necessari alla nuova compilazione del nostro file `.tex` .
Occorre infatti ricompilare il nostro file per ottenere gli effetti desiderati.

Naturalmente il processo di compilazione, creazione dei thumbnails e ricompilazione può essere automatizzato mediante un `Makefile` come quello incluso nel pacchetto di sorgenti del presente Howto oppure mediante un semplice script come il seguente.

```
#!/bin/bash

function makepdf () {
#NOME=howto
pdflatex $NOME.tex
pdflatex $NOME.tex
pdflatex $NOME.tex
thumbpdf --verbose $NOME.pdf
pdflatex $NOME.tex
}

echo "Inserisci il nome del file .tex senza estensione!"
read NOME
if [ -e $NOME.tex ]
then
  makepdf
else
  echo "Il file non esiste"
fi
```

Purtroppo alcuni programmi potrebbero non offrire la visualizzazione dei thumbnails, pur non inficiando assolutamente la corretta visualizzazione del corpo del documento. Ci sembra tuttavia che Acrobat Reader (sia su piattaforma Unix che Windows) non abbia di queste mancanze.

7.2 ... usando `dvipdfm`

Teoricamente è sufficiente utilizzare il comando `dvipdft` invece di `dvipdfm`; `dvipdft` è uno script di shell `sh` che "intelligentemente" provvede alla preparazione dei thumbnails (sempre utilizzando GhostScript) e alla compilazione in modo tale da fare tutto il necessario. Tuttavia abbiamo riscontrato che sulle nostre macchine non riesce a produrre i thumbnails (GhostScript segnala un errore) a causa del fatto che la shell interpreta un carattere "`%`" in maniera diversa da come desidera l'autore. Per questo motivo, laddove l'uso di `dvipdft` dovesse presentare tale problema, si suggerisce di preparare il `.pdf` senza thumbnails con il comando `dvipdfm` (analogamente a come si fa con `pdflatex`) e quindi produrre i thumbnails utilizzando il seguente script, che usa lo stesso comando di base ma fa riferimento alla shell `csh` anziché `sh` e sulle nostre macchine non presenta il problema in questione:

```
mythumbs
-----------------------------------------------
```

```
#!/bin/csh

gs -r10 -dNOPAUSE -dBATCH -sDEVICE=png256 -sOutputFile=$1.%d $1.pdf
-------------------------------------------------
```

Tale semplice script può essere usato con il comando

```
./mythumbs [nomefile]
```

oppure

```
csh mythumbs [nomefile]
```

indicando il `[nomefile]` senza estensione.
Prodotti i thumbnails, è sufficiente usare il comando

```
dvipdfm -t [nomefile].dvi
```

dove l'opzione "`-t`" serve a includere i thumbnails nel documento. Usando anche l'opzione "`-d`" si fa in modo che i thumbnails vengano rimossi alla fine della compilazione:

```
dvipdfm -t -d [nomefile].dvi
```

8 Un utile consiglio...

Nei manuali LaTeX che ci è capitato di leggere (anche su Appunti di Informatica Libera) veniva spesso introdotto un preambolo che induce all'utilizzo del package `fontenc` con la codifica `T1` dei caratteri, mediante questa riga di codice inclusa nel preambolo:

```
usepackage[T1]{fontenc}
```

Ebbene l'utilizzo di questo package può rendere letteralmente disastrosa la resa visiva del file PDF qualora si intervenisse con `pdflatex`. Questo avviene perché l'inclusione del package sopracitato comporta l'utilizzo di fonts non scalabili con conseguente risultato "sfocato" nel file PDF (vedere [3] a proposito dei font EC, che non sono liberamente utilizzabili in formato "Type 1"). Ci raccomandiamo di non utilizzare GhostView per visualizzare il prodotto della compilazione. GV, infatti, è un prodotto piuttosto raffinato che tende a coprire alcuni difetti grafici dei documenti mediante l'applicazione dell'anti-aliasing. Consigliamo dunque di usare, almeno per una verifica, strumenti semplici come `xpdf`.
Ulteriori informazioni su questo punto sono reperibili in [3], nella sezione relativa alla produzione di documenti PDF.

9 Preparazione e gestione delle immagini

In questa sezione vengono presentati alcuni programmi per la produzione e la manipolazione di immagini da usare per la produzione di documenti PDF mediante LaTeX. Sono considerati sia programmi il cui uso è a linea di comando (che quindi permettono di sfruttare la potenza e la versatilità della shell di Unix/Linux) che programmi dotati di interfaccia grafica.

9.1 ImageMagick

Programma molto interessante e potente, disponibile per molte piattaforme, tra cui praticamente tutti i sistemi operativi Unix (compreso Linux) e l'ambiente MS Windows. È utilizzabile sia a linea di comando che con interfaccia grafica. In particolare, se lo si vuole usare per convertire immagini da un formato a un altro (e magari anche per fare delle elaborazioni delle stesse immagini), si suggerisce di utilizzarne i comandi `convert` e `mogrify`, che permettono di fare un gran numero di operazioni e per l'uso dei quali si rimanda alle relative pagine di manuale.

9.2 netpbm

`netpbm` è utilizzabile soprattutto su sistemi Unix ed è un pacchetto molto utile per la manipolazione di immagini da linea di comando; comprende moltissimi comandi che permettono, tra le altre cose, di effettuare conversioni su un grandissimo numero di formati grafici.
Di seguito sono illustrate le conversioni da BMP e da JPEG a PNG e viceversa. In questa sezione, l'asterisco è utilizzato per brevità e ad esso va sostituito il nome del file, a meno di diverso avviso.

9.2.1 Da BMP a PNG

La conversione di un'immagine `*.bmp` può essere effettuata attraverso una serie di passaggi intermedi che la riga di comando permette di condensare facilmente in un'unica stringa. Verrà prima convertita l'immagine `*.bmp` in una `*.ppm` (portable pixmap), quindi la `*.ppm` nell'immagine finale `*.png`. Ci appoggeremo quindi ai seguenti comandi:

```
bmptoppm
pnmtopng
```

Otterremo quindi il `*.png` con il comando

```
bmptoppm [nomefile].bmp | pnmtopng > [nomefile].png
```

9.2.2 Da PNG a BMP

Sostanzialmente si tratta di percorrere il processo già effettuato usando questa volta i seguenti comandi:

```
pngtopnm
ppmtobmp
```

Otterremo quindi il `*.bmp` digitando:

```
pngtopnm [nomefile].png | ppmtobmp > [nomefile].bmp
```

Il procedimento non cambia per gli altri formati, se non per il nome dei comandi, quindi saremo più brevi.

9.2.3 Da JPEG a PNG

```
jpegtopnm [nomefile].jpg | pnmtopng > [nomefile].png
```

9.2.4 Da PNG a JPEG

```
pngtopnm [nomefile].png | ppmtojpeg > [nomefile].jpg
```

9.2.5 Conversioni di altri formati

Attraverso metodi molto simili è possibile convertire in png ed in molti altri file di estensioni diverse (gif, tiff, tga, etc...) molti tipi di immagini. Inutile dunque elencare tutte le combinazioni possibili. I comandi di conversione possono essere reperiti molto semplicemente. In pratica basta digitare l'estesione che ci interessa, quella di partenza, e premere tab 2 volte. Tra le possibili opzioni presentate ci sarà certamente quella che ci interessa ossia quella che consente di passare ad un file con estensione `.ppm` o `.pnm`. La conversione tramite pipeline è quindi già stata chiarita e può essere utilizzata senza problemi. Per conoscere in che formati si può convertire ad esemipio un file `.ppm` basta digitare "`ppm`", premere due volte tab, ed analizzare il risultato. Oppure si può usare il seguente comando:

```
locate ppm | grep ppmto
```

Niente di troppo complesso.

9.3 libjpeg

`libjpeg` è un altro pacchetto che può essere utile per la conversione delle immagini da linea di comando.

9.4 `ps2pdf` e `epstopdf`

Disponibili per le piattaforme: Unix, Windows

Sono due programmi utili a convertire figure sia PS che EPS in PDF. Poiché su Unix è facile produrre un output PS da praticamente tutti gli applicativi e su Windows per farlo è sufficiente installare una stampante PS (si può usare uno qualsiasi dei driver di stampante PostScript disponibili sul CD di Windows) e stampare su file, la possibilità di convertire in PDF le figure PS ed EPS è particolarmente interessante.

Il primo, `ps2pdf`, fa parte di GhostScript, mentre il secondo, `epstopdf`, è incluso nel teTeX, cioè nel Thomas Esser TeX [3], che probabilmente è la distribuzione LaTeX più usata su piattaforma Unix. Nel teTeX, `epstopdf` è uno script Perl; quindi, oltre che di GhostScript, fa uso di Perl; rispetto a `ps2pdf` ha in più la capacità di riuscire a individuare correttamente in moltissimi casi il `BoundingBox` della figura che converte, con ovvi vantaggi in termini di correttezza dei confini dell'immagine che si ottiene a valle della conversione (il `BoundingBox` individua le coordinate dei bordi della figura e, quindi, la posizione relativa degli oggetti rispetto ai bordi e le dimensioni orizzontale e verticale della figura; visualizzando il contenuto ASCII di una delle figure PS/EPS incluse nel presente documento, è immediato notare il `BoundingBox` definito tra le prime righe). Per questo motivo, si suggerisce di preferire `epstopdf` a `ps2pdf`. Se la figura PS/EPS da convertire è stata ottenuta da un applicativo di Microsoft Office, molto probabilmente il `BoundingBox` risultante è abbondante; i "perfezionisti" possono modificarlo correggendone con un qualunque editor le coordinate direttamente nel file PS/EPS ottenuto; le coordinate giuste sono facilmente individuabili aprendo il file con un lettore PS che mostri le coordinate su cui passa il mouse mentre si trova sulla figura.

A chi non ha pazienza e tempo e voglia per fare tutto ciò, suggeriamo di ricorrere a Gimp, considerato più avanti.

Per ciò che riguarda l'ambiente Windows, le versioni recenti di Miktex includono un eseguibile che serve a svolgere lo stesso compito dello script Perl `epstopdf` incluso nel teTeX, senza bisogno di installare Perl, dato che si tratta di un compilato di un programmino scritto in C.

9.5 The GIMP

Disponibile per le piattaforme: Unix, Windows (la versione per Windows è un porting con meno funzionalità e non è stabile quanto quella per Unix).

The GNU Image Manipulation Program, ottimo programma per l'elaborazione grafica, è molto utile nel presente contesto, sia perché permette di creare e di ritoccare immagini, sia perché è in grado di leggere immagini in vari formati, vettoriali e non, e di salvare in moltissimi formati grafici, tra cui alcuni di interesse per la produzione di documenti PDF tramite LaTeX: PNG, Jpeg. In particolare, Gimp importa anche file in formato PS, EPS o PDF. Quindi Gimp risulta utile per salvare in PNG un'immagine PS, EPS o PDF i cui confini non sono opportuni: basta importarla, ritagliare la parte desiderata e salvarla come immagine PNG. Inoltre, Gimp permette di catturare degli screenshot dello schermo o di una finestra; quindi, nella peggiore delle ipotesi, se l'immagine da utilizzare è stata realizzata con un applicativo a partire dal quale è difficile ottenere un'immagine in un formato utile al contesto qui considerato, è sufficiente visualizzare l'immagine sullo schermo mediante l'applicativo in questione e poi catturare uno screenshot della visualizzazione sullo schermo dell'immagine, per poi salvare lo screenshot in formato utile, ad es., PNG. Essendo disponibile anche per Windows, può essere facilmente usato per catturare screenshots anche dagli applicativi di Microsoft Office.

Inoltre Gimp è adatto per interfacciarsi con gli scanner, usando Sane/XSane su Unix e l'interfaccia Twain su Windows (praticamente fa tutto da solo...).
URL di riferimento: `http://www.gimp.org/`

9.6 XFig

Disponibile per piattaforme Unix; esiste una versione per Windows, che però richiede l'utilizzo di altro software, tra cui un server grafico X-Window, cioè dello stesso tipo usato su Unix.

XFig è un programma molto valido per produrre figure con grafica vettoriale; permette di esportare le figure nei formati PS, EPS, PDF (e non solo). Le versioni più vecchie non esportano in PDF, ma, a partire dal formato PS/EPS, si può ottenere una figura in PDF usando `epstopdf`. Le figure prodotte non utilizzano font non scalabili che provocherebbero i problemi citati a proposito di `ps2pdf`. XFig permette di salvare anche in formati grafici come ad esempio GIF, JPG e PNG, ma, per questi scopi, probabilmente si può ottenere un'immagine di qualità migliore salvando come PS/EPS e gestendo il risultato con Gimp.
URL di riferimento: `http://www.xfig.org/`

9.7 Tgif

Dal punto di vista di questo Howto, Tgif è un programma analogo a XFig.

URL di riferimento: `http://bourbon.usc.edu:8001/tgif/`

9.8 Gnuplot

Disponibile per piattaforme Unix, Windows.

Gnuplot è un programma orientato alla produzione di grafici, sia 2D che 3D. Può essere utilizzato sia interattivamente da linea di comando che in modalità script/batch, cioè mediante degli script contenenti comandi nel suo linguaggio. Un po' ostico inizialmente, ma molto comodo e potente. Oltre all'output (`terminal`) grafico usuale sullo schermo, prevede l'output su file in moltissimi formati diversi, tra cui PS ed EPS (direttamente utilizzabili, sulla base delle considerazioni già fatte) e il formato FIG, utilizzato da XFig. Riguardo a quest'ultimo formato, la possibilità di uscire in FIG è molto utile perché permette di modificare visualmente mediante l'interfaccia grafica di XFig il risultato ottenuto e infine produrre una figura nel formato desiderato, sulla base delle considerazioni fatte riguardo a XFig. A Gnuplot si appoggiano, per un'uscita grafica, programmi matematici come Octave. Le figure prodotte non utilizzano font non scalabili che provocherebbero i problemi citati a proposito di `ps2pdf`.
URL di riferimento: `http://www.gnuplot.info/`

9.9 Gnuplot plus

Gnuplot plus è una "patch" (aggiunta e modifiche) a Gnuplot che serve a offrire funzionalità addizionali:

1. la possibilità di utilizzare comandi di tipo LaTeX quando si usa il terminale PostScript
2. gestione di testo giapponese
3. salvataggio della storia dei comandi mediante l'utilizzo della libreria GNU readline (su Unix/Linux)
4. possibilità di avere l'help in lingue diverse sulla base della variabile d'ambiente `LANG`

Il punto 1 è molto interessante in questo contesto, dato che risulta possibile (con determinate limitazioni) inserire ad es. equazioni LaTeX nella composizione di un grafico. Per sfruttare tale possibilità, bisogna utilizzare il terminale `postscript plus`.

In Fig. 3 si riporta un grafico di esempio prodotto appunto con Gnuplot Plus. Il codice corrispondente a tale figura è:

```
sinus.plt
-------------------------------------------------

#!/usr/bin/gnuplot

set size 1.5,0.7
set term postscript portrait plus 'Times-Roman' 18
set output "sinus.ps"

set grid
```

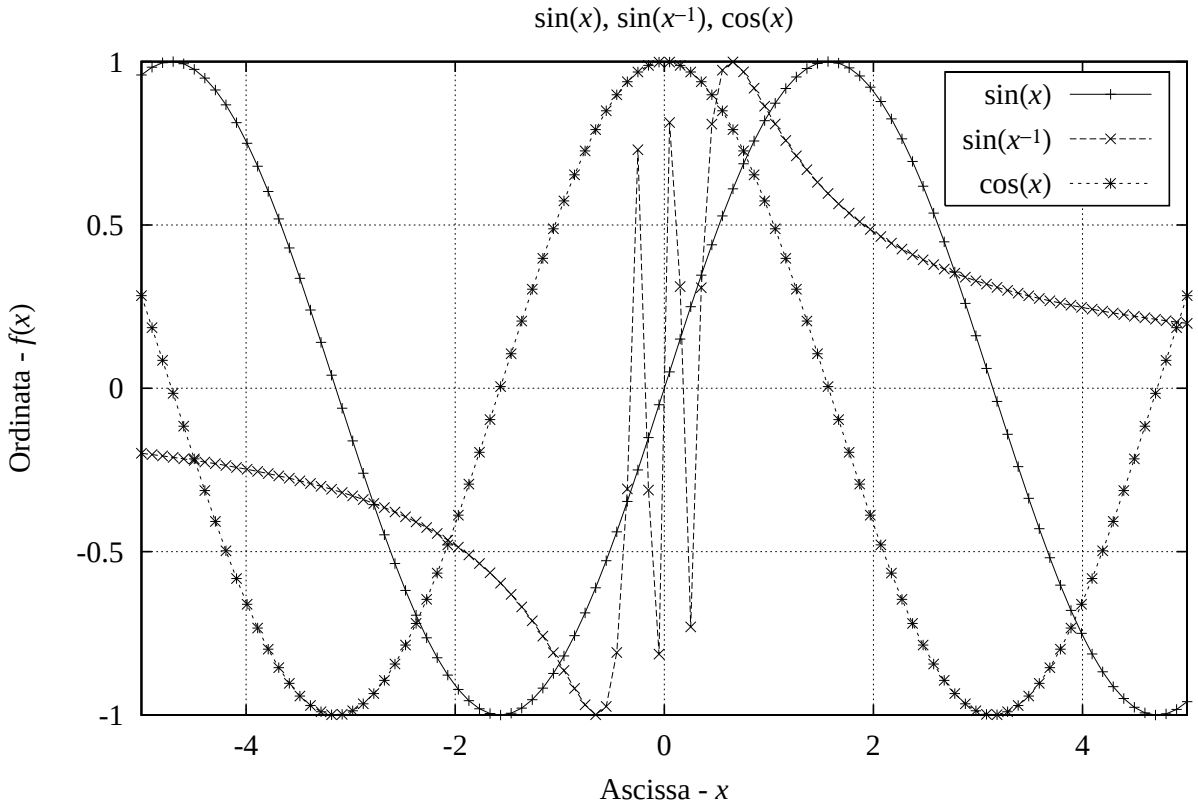

Figura 3: Esempio di grafico prodotto con Gnuplot Plus

```
set xrange [-5:5]
set yrange [-1:1]

set title "sin$(x)$, sin$(x^{-1})$, cos$(x)$"
set xlabel "Ascissa - $x$"
set ylabel "Ordinata - $f(x)$"
set key right top spacing 1.5 width -7 box

plot \
        sin(x) title "sin$(x)$" with linespoints, \
        sin(1/x) title "sin$(x^{-1})$" with linespoints , \
        cos(x) title "cos$(x)$" with linespoints
```

URL di riferimento:
`http://www.yama-ga.com/gnuplot/`

9.10 OpenOffice.org

Disponibile per le piattaforme: Unix, Windows.

OpenOffice.org è una suite Open Source di office automation molto interessante e multipiattaforma, la cui versione commerciale, cioè Star Office, è supportata commercialmente

da SUN Microsystems. Offre funzionalità analoghe a Microsoft Office e filtri di importazione/esportazione per garantire una buona interoperabilità con gli utenti di Microsoft Office. I suoi moduli (word processor, presentation chart, spreadsheet, ecc.) possono essere comodamente usati per preparare figure/grafici da inserire in documenti LaTeX. Da OpenOffice.org risulta semplicissimo stampare su file in formato PS. Un'altra possibilità molto comoda: OpenOffice.org permette anche di stampare solo la selezione corrente, ignorando tutti gli altri oggetti del documento corrente. Una cosa un po' fastidiosa è che attualmente OpenOffice.org stampa su pagine nei formati standard ma non su un file encapsulated postscript che sia senza formato di pagina e abbia solo dei confini individuati dalle coordinate del proprio `BoundingBox`. Quindi può risultare necessario modificare "manualmente" il file postscript ottenuto stampando su file, per correggerne il `BoundingBox`, facendo ad esempio come indicato più avanti. Nel caso si dovessero incontrare problemi nel fare tale modifica (o non si volesse fare tale modifica "manuale"), si suggerisce di uscire comunque in PS ma di utilizzare Gimp per importare il PS ottenuto e salvare in un formato grafico comodo (PNG, Jpeg) solo la parte di immagine di interesse. Si sconsiglia invece di usare Gimp per catturare uno screenshot del documento aperto in OpenOffice.org, dato che le opzioni di importazione di un file PS mediante Gimp permettono di agire in maniera molto più fine e mirata e di ottenere un risultato migliore.

Di seguito, un esempio di come modificare il `BoundingBox` del postscript ottenuto stampando su file. Si suggerisce di usare il presentation chart, cioè OpenOffice.org Impress, con orientazione di pagina portrait (non landscape) e di stampare su file scegliendo l'orientazione portrait e ad esempio il formato di pagina A4. Quindi si apre il postscript con gv e si scorre con il mouse sull'immagine; le coordinate del puntatore del mouse sull'immagine sono continuamente rilevate e mostrate da gv in una apposita parte della sua interfaccia; scorrendo con il mouse, si individuano le coordinate `(x1,y1)` e `(x2,y2)` che si vuole facciano da confine all'immagine, facendo riferimento al seguente schema:

```
   +------------(x2,y2)
   |               |
   |               |
   |               |
   |               |
(x1,y1)------------+
```

Quindi si apre il postscript con un editor di testo (ad esempio, VIM), si cercano le righe contenenti `BoundingBox` e `PageBoundingBox` e su tali righe si sostituiscono i 4 numeri indicati dopo i due punti con `x1 y1 x2 y2`. Infine, all'interno del postscript, si cerca `BeginFeature: *PageSize` e si rimuovono le righe tra `BeginFeature` e `EndFeature`, comprese le righe contenenti tali stringhe. A questo punto si ricarica il postscript con gv e come confini si dovrebbero notare proprio quelli voluti; da questo punto in poi, la figura postscript può essere usata come già esposto, ad esempio può essere convertita usando `epstopdf`.
URL di riferimento: `http://www.openoffice.org/`

9.11 Altri programmi

Si citano infine:

- DIA (Gtk+, Gnome)

- Gnumeric (Gnome)
- Koffice (KDE)
- Grace - `http://plasma-gate.weizmann.ac.il/Grace/`

10 Come preparare delle slide PDF con LaTeX

I contenuti già esposti si applicano anche alla classe `slide`, quindi sono validi anche per la produzione di semplici slide. Tuttavia, per produrre slide con LaTeX esistono anche pacchetti più potenti, come:

- `seminar` (già incluso nel teTeX),
- `lucassem`
  `http://l3.web.cern.ch/l3/lucassem/lucassem.html`
- `ifmslide`
  `http://www.ctan.org/tex-archive/help/Catalogue/entries/ifmslide.html`
  che richiede anche texpower
  `http://texpower.sourceforge.net/`
- `pdfslide`
  `http://www.ctan.org/tex-archive/help/Catalogue/entries/pdfslide.html`
- `prosper` (incluso in Debian Woody)
  `http://www.ctan.org/tex-archive/help/Catalogue/entries/prosper.html`
  `http://prosper.sourceforge.net/`
- `slidenotes`
  `http://www.ctan.org/tex-archive/help/Catalogue/entries/slidenotes.html`

Di seguito si prendono in considerazione l'uso di `lucassem` con `dvipdfm` e di `ifmslide` con `pdflatex` e si forniscono dei semplici esempi orientati soprattutto alla presentazione di contenuti scientifici, come ad esempio quelli che possono essere oggetto di una lezione o di un seminario nell'ambito di Matematica, Fisica, Ingegneria; per usi diversi e/o esigenze più complesse, si rimanda alla documentazione dei pacchetti già citati.
Inoltre, si cita un interessante tutorial su `prosper`.

10.1 `lucassem + dvipdfm`

Pacchetto interessante, piuttosto comodo e potente. Pensato per la produzione di slide in postscript, non è utilizzabile con `pdflatex`, ma è possibile usarlo con `dvipdfm` rinunciando ad alcune sue potenzialità. Nell'esempio allegato è stata inserita una versione leggermente personalizzata di `lucassem.cls`, cioè `mplucassem.cls` (`mp` sta per Marco Pratesi :-). Come si può leggere nel preambolo dell'esempio allegato, il driver `dvipdfm` non può essere caricato nel modo usuale, come esposto precedentemente, ed è necessario includere `dvipdfm.def` in maniera un pochino "brutale":

```
\makeatletter
\input dvipdfm.def
\makeatother
```

Quando si usa `lucassem` con `dvipdfm`, purtroppo si deve rinunciare a produrre delle slide portrait, che invece possono essere comodamente ottenute se si usa `lucassem` per produrre slide postscript. Questo è dovuto al fatto che con questo pacchetto le slide portrait vengono ottenute usando direttive special riguardanti proprio l'uso del formato postscript, che sono comprensibili per `dvips` ma non per `dvipdfm`:

```
ps: tx@Dict begin 90 Rot  end
...
ps: tx@Dict begin -90 Rot  end
```

Questo è un limite spiacevole, ma solo in alcuni casi: se le slide devono essere stampate, si può tranquillamente far uso del formato postscript (per la produzione del quale, tale limite non c'è); se invece vanno proiettate usandole in formato elettronico, allora il portrait viene comunque evitato praticamente sempre, dato che non offre una buona resa; in definitiva, tale limite si presenta concretamente solo se le slide devono essere sia stampate che prodotte in formato PDF. Inoltre, si ricorda che si può anche usare tale pacchetto per produrre delle slide postscript e poi convertire queste ultime in PDF usando `psp2df`; in tal caso, è bene procedere come indicato nella sezione dedicata a `ps2pdf`, per fare in modo che dvips usi font "Type 1", scalabili. L'esempio allegato comprende un `Makefile` che prevede la produzione delle slide in formato postscript ("`make`" o "`make ps`"), pdf ("`make pdf`"), postscript con due slide per pagina ("`make compact`"). Si fa notare che è possibile passare rapidamente e comodamente da una versione a colori a una versione in bianco e nero delle stesse slide semplicemente scommentando la riga contenente il comando

```
\blackandwhite
```

10.2 ifmslide + pdflatex

Pacchetto molto interessante e potente, concepito per la produzione di slide in pdf (ma non solo in pdf, anche in dvi e postscript). Permette di realizzare documenti ben più complessi e ricchi rispetto all'esempio allegato, che, come già scritto, si rivolge fondamentalmente a chi deve fare una presentazione o un seminario/lezione in ambito scientifico. Permette senz'altro di realizzare documenti contenenti sia slide landscape che slide portrait. Nell'esempio allegato si propone un modo piuttosto semplice ed elementare per inserire delle slide portrait nel documento e per usare degli header e footer piuttosto semplici.
Nota: compilando su Debian 3.0 "Woody" l'esempio allegato, si riscontra un errore corrispondente al seguente baco noto:
`http://bugs.debian.org/cgi-bin/bugreport.cgi?bug=165732&repeatmerged=ye`
Un workaround molto banale corrisponde all'evitare l'inclusione di packages di cui in effetti l'esempio non fa uso, cioè `amssymb`, `amstext`, `babel`. Diversamente, all'URL indicata sopra è spiegato come correggere il baco in questione; una soluzione sporca e veloce: modificare `/etc/texmf/texmf.cnf` commentando le righe

```
pool_size.context = 750000
pool_size = 125000
```

e aggiungendo le righe

```
pool_size.context = 3000000
pool_size = 500000
```

10.3 `prosper`

Su Freshmeat, all'URL `http://freshmeat.net/articles/view/667/` si può trovare un interessante tutorial sull'uso di `prosper`: "*Making Presentations with LaTeX and Prosper*", di Rajarshi Guha.

11 Ringraziamenti

Desideriamo ringraziare l'Ing. Pierpaolo Vittorini per aver fornito importanti elementi relativi alla sezione su `dvipdfm`, Michele Antonecchia e il Dott. Marco Presi per il lavoro di revisione del presente documento e per i suggerimenti forniti.

Riferimenti bibliografici

[1] Tobias Oetiker, Hubert Partl, Irene Hyna, and Elisabeth Schlegl, *The Not So Short Introduction to LaTeX 2ε*, tradotto anche in italiano, disponibile nella distribuzione teTeX; su Red Hat Linux 7.2, 7.3 e 8.0 (pacchetto `tetex-doc`): `/usr/share/texmf/doc/latex/general/lshort.dvi` ; su Debian 3.0 "Woody" (pacchetto `tetex-doc`): `/usr/share/doc/texmf/latex/general/lshort.dvi.gz`
Un URL di riferimento:
`ftp://ftp.uniroma2.it/TeX/info/lshort/`

[2] Marc Baudoin, *Apprends LaTeX*, tradotto in italiano (*Impara LaTeX*) da Alessandro Cannarsi.
Alcuni URL di riferimento:
`ftp://ftp.univ-lyon1.fr/pub/doc/francais/texte/apprends_latex/`
`http://ftp.pluto.linux.it/pub/pluto/ildp/misc/impara_latex/`

[3] Thomas Esser, *teTeX Manual*, disponibile nella distribuzione teTeX; il file di riferimento è `TETEXDOC.tex`; su Red Hat Linux 7.2, 7.3 e 8.0 (pacchetto `tetex-doc`): `/usr/share/texmf/doc/tetex/TETEXDOC.*` ; su Debian 3.0 "Woody" (pacchetto `tetex-base`): `/usr/share/doc/texmf/tetex/TETEXDOC.*`

[4] The Comprehensive TeX Archive Network (CTAN):
`http://www.ctan.org/`

[5] Pagina di ricerca del CTAN:
`http://www.ctan.org/search/`

12 Quick Guide: i comandi a portata di mano

Questa sezione è dedicata ad una guida rapida e maneggevole che riassume il contenuto dell'howto. Pertanto è povera di spiegazioni, per le quali si rimanda ovviamente all'howto stesso.

12.1 `ps2pdf`: la strada più veloce

Comando: `ps2pdf [nomefile].ps`

Formato A4: Modificare lo script `ps2pdfwr` ed inserire `-sPAPERSIZE=a4`
nella riga di esecuzione di `gs`

```
exec gs $OPTIONS -q -dNOPAUSE -dBATCH -sDEVICE=pdfwrite
  -sPAPERSIZE=a4 -sOutputFile=$outfile $OPTIONS -c
  .setpdfwrite -f $infile
```

Fonts Type1: **teTeX**
Creare il file `.dvipsrc` nella propria home ed inserirvi le seguenti righe:

```
p +psfonts.cmz
p +psfonts.amz
```

MikTeX
Aggiornare il file `\texmf\dvips\config\config.ps`

Alternative: **Utilizzare altri tipi di fonts. Ad esempio i `Times`**
`\usepackage{times}`
`\usepackage{mathptm}` (per l'ambiente matematico)

Note: È possibile utilizzare gli script `ps2pdf12`, `ps2pdf13` e `ps2pdf14`
al posto di `ps2pdf` per generare dei pdf nel rispettivo formato.
Di default viene utilizzato `ps2pdf12`.
Dal `.dvi` al `.pdf` in un sol colpo

```
dvipdf [nomefile].dvi
```

Attenzione al comando `\usepackage[T1]{fontenc}`

12.2 pdflatex e dvipdfm: pensare in pdf

Un preambolo universale:

```
01 \ifx\pdfoutput\undefined % We are not running pdftex
02 \documentclass[12pt,a4paper]{article}
03 \usepackage[dvips]{graphicx}
04 %\usepackage{psfig}
05 %\usepackage{epsfig}
06 \RequirePackage[dvipdfm,hyperindex]{hyperref}
07 %\RequirePackage[dvipdfm,colorlinks,hyperindex]{hyperref}
08 \else
09 \documentclass[pdftex,12pt,a4paper]{article}
10 \usepackage[pdftex]{graphicx}
11 \DeclareGraphicsExtensions{.pdf,.png,.jpg,.mps}
12 \RequirePackage[hyperindex]{hyperref}
13 %\RequirePackage[colorlinks,hyperindex]{hyperref}
14 \usepackage{thumbpdf}
15 \fi

\hypersetup{
pdftitle={LaTeX-PDF-Howto},
pdfsubject={Come produrre documenti PDF con LaTeX},
pdfauthor={Marco Pratesi, Marco Latini},
pdfkeywords={LaTeX, PDF, Howto, pdflatex, dvipdfm},
%pdfpagemode=FullScreen
}
```

Inclusione di una figura:

```
01 \begin{figure}
02 \centerline{\includegraphics
      [width=0.7\textwidth]{category}}
03 \caption{Categories of Free and Non-Free
      Software, taken from
04 {\tt http://www.gnu.org/philosophy/categories.html}}
05 \label{software-categories}
06 \end{figure}
```

Solo dvipdfm: Se non si ha interesse a generare il documento postscript si può evitare di creare le immagini in entrambi i formati ed utilizzare soltanto quelli utili per il pdf. Ad esempio il formato PNG.
Innanzitutto nel preambolo bisogna sostituire la riga 02 con:
`\documentclass[dvipdfm,12pt,a4paper]{article}`
Creare il Bounding Box alle figure:
`ebb linus3.png` (per tutte le figure non ps/eps)
Procedere normalmente con la compilazione.

Comandi: **pdflatex**

```
pdflatex [nomefile].tex
```
(tre volte)

dvipdfm

```
latex [nomefile].tex
```
(tre volte)

```
dvipdfm [nomefile].dvi
```

Thumbnails: **pdflatex**

```
pdflatex [nomefile].tex
```
(tre volte)

```
thumbpdf [nomefile].pdf
pdflatex [nomefile].tex
```

dvipdfm

```
latex [nomefile].tex
```
(tre volte)

```
gs -r10 -dNOPAUSE -dBATCH -sDEVICE=png256
  -sOutputFile=[nomefile].%d [nomefile].pdf
dvipdfm -t -d [nomefile].dvi
```