Introducción a LaTeX para tipografía de textos científicos y técnicos

José M.ª Goicolea, Felipe Gabaldón, Luis Seidel, Santiago Muelas
Universidad Politécnica de Madrid

Sesión 3. 16 de febrero de 2000

Sumario

Apología

— Con TEX/ LATEX se puede hacer **todo**

— Para mantener: software libre y portable

— Para aprender: software singular y venerable

— «Obras maestras del arte tipográfico» frente a documentos *basura*
(30.9.99 comp.text.tex) Alguien dice: «`At the moment I am using Word97. The layout is ugly, but it IS convenient.`»
S. Rahtz contesta: «`so are McDonalds burger shops. both of them poison you.`»

— With a little help from my friends

El impresor toma el mando: de DVI a una salida gráfica

TEX es el cajista: con unas reglas tipográficas muy exigentes, llena cada página de cajas alineadas que contendrán las letras. Todo lo que necesita para generar el DVI es uno o varios archivos `.tfm` (TEX font metrics).

Un archivo DVI (DeVice Independent) es verdaderamente independiente del dispositivo que vayamos a utilizar para obtener el documento impreso. Para la salida final necesitamos otro programa independiente de TEX: el manipulador de DVI (*DVI-driver*). Al menos la mitad de la responsabilidad de obtener un bonito documento cae sobre él.

En los primeros tiempos, éste servía para pasar de DVI a impresora o pantalla (YAP hace eso). Ahora puede ser un puente a un formato estándar (PS o PDF).

Los `\special`'s: Knuth dejó una primitiva que era ignorada por TEX, dejaba huella en el DVI y debía ser interpretada por el *DVI-driver*: tenemos un DVI no «portable». Es lo que hay debajo del soporte a gráficos, color, hipertexto y etiquetas en el archivo fuente.

De DVI a PostScript: dvips

El *DVI-driver* para convertir a PostScript es dvips, escrito por Tomas Rokicki: presume de generar PostScript excelente. Está en la versión 5.86. *Software libre*.

PostScript es el lenguaje de descripción de páginas desarrollado por Adobe que es estándar en la industria gráfica.

El intérprete GhostScript, y el visualizador GSView (PS, PDF).

El BoundingBox

Al ser un lenguaje de programación, con un archivo PS (que además es ASCII) se puede hacer casi de todo: puede incluir tipos vectoriales (independientes de la resolución).

psutils: la mejor forma de reordenar páginas:

```
psnup -2 -r -m3cm uno.ps kkseidel.ps
```

De DVI a PDF: dvipdfm

El *DVI-driver* para convertir a PDF es `dvipdfm`, escrito por Mark A. Wicks. Está en la versión 0.12.7b. *Software libre*.

PDF (Portable Document Format) es un formato estándar desarrollado por Adobe a partir de PostScript, sin capacidades de programación pero con toda la capacidad tipográfica, posibilidades de hipertexto, formularios, multimedia,... Es un formato ideal para la distribución electrónica (o en la web) de documentos complejos y bien terminados.

Argumentos puristas de dvipdfm

TEX es a Postscript, como DVI es a PDF. DVI y PDF describen la página.

`dvipdfm -p a4 -l kkseidel.dvi`
`dvipdft -p a4 -l kkseidel` (necesita Ghostscript).

Todos los caminos llevan a PDF

En el camino `.tex` $\longrightarrow$ `.pdf`, podemos seguir varias rutas.

pdftex Variante de TeX para pasar directamente de `.tex` a `.pdf`.

dvipdfm De `.tex` a `.dvi` con LaTeX y de `.dvi` a `.pdf` con dvipdfm.

Distiller De `.tex` a `.dvi` con LaTeX, de `.dvi` a `.ps` con dvips y de `.ps` a `.pdf` con Distiller de Adobe (comercial) o pdfwrite (incluido en Ghostscript, bueno a partir de la versión 6).

Algunas de las posibilidades de PDF

Una calculadora

Ejercicios interactivos

Poster de Congresos

Esos tipos peligrosos

Tipos de imprenta: las fuentes son para calmar la sed.

TEX es singular: Knuth no sólo escribió TEX , sino que le dió una hermana (MetaFont) y una familia (de tipos): Computer Modern. Durante mucho tiempo han sido la «marca de fábrica» de los documentos escritos en TEX.

En LATEX 2.09 (antes de 1994) era dificil utilizar otras familias de tipos (Times, Helvética, Palatino,...). En LATEX 2_ε tenemos NFSS (New Font Selection Scheme), que facilita la tarea.

En LATEX un tipo (para texto) se caracteriza por cinco atributos: **codificación** (OT1, T1, OMS), **familia** (Computer Modern, Adobe Times), **serie** (lo ancho que es un tipo), **forma** (recta, cursiva, inclinada) y **tamaño** (para el que ha sido diseñado, 10pt, 12pt). Cada combinación debe dar lugar a un único archivo `.tfm`. Queda una huella en el `.log`

```
LaTeX Font Info:    Font shape ‘OT1/fcmtt/bx/n’ in size <24.88> not available
(Font)              Font shape ‘OT1/fcmtt/m/n’ tried instead on input line 76.
```

Puedo seleccionar una combinación con `\usefont{T1}{ptm}{b}{it}` y ver qué pasa:

`\usefont{T1}{ptm}{m}{it}`	*Ahora escribo con Times Cursiva.*
`\usefont{T1}{phv}{m}{n}`	Ahora con Helvética.
`\usefont{T1}{pcr}{m}{sl}`	*Ahora con Courier inclinada.*

Para volver a la «normalidad»: `\normalfont`

Estilos precocinados:

Que cambian el tipo base:	`avant, chancery, charter, courier,helvet,utopia`
Que cambian los tres:	`bookman, newcent, palatino,palatcm,times`
Que cambian los tipos matemáticos:	`mathptm,mathptmx`
Para dingbats:	`pifont`

El problema del €

ASCII, Ansinew (Windows, cp1252), iso-latin-1, iso-latin-15, Unicode.

Primero el teclado o un mandato, luego un `.tfm`, luego un tipo `.pk` o `.pfb` con una determinada codificación.

Con nuestra ñ, lo mismo; solución inicial `\usepackage[T1]{fontenc}`.

Pega importante: no hay tipos EC vectoriales (Type1).

¿Volvemos a OT1? Entonces no parte las palabras acentuadas ni las trata bien en PDF (buscar).

¿Entonces? `\usepackage{ae}` (Almost European). Pero no viene en mikTEX ni incluye las comillas tipográficas.!!!!!!!

El problema del €: una solución

```
\documentclass{minimal}
\usepackage{marvosym}
\begin{document}
\thispagestyle{empty}
\EUR
\end{document}
```

Lo convierto en eps (`dvips -E`) o en pdf y lo incluyo

```
\includegraphics[width=0.9em]{euro.eps}
```

Esto vale 999€

Esto vale 999€

No es tan chapuza como parece...

¿Dónde coloco las fotos del verano o las superficies equipotenciales?

TEX ignora los gráficos.

LATEX se contenta con reservar una caja de tamaño adecuado en un sitio adecuado.

De nuevo, *casi todo* depende de cómo manipulemos el DVI. Al menos hay una interfaz consistente y común para todos los *drivers*

`\usepackage[driver]{graphicx}`, en el preámbulo.
`\includegraphics[opciones]{archivo.eps}` en el punto donde quiero la caja.

Para muchas más opciones, ver `C:\texmf\doc\latex\graphics\epslatex.ps`

Como con los cuadros, se logra una mejor composición tipográfica dejando que LATEX decida como colocar el gráfico poniéndolo en un entorno `figure`.

Ejemplo (ilegal)

```
\begin{figure}[h!]
\centering
\includegraphics[width=0.3\textwidth]{Brir12.jpg}
\caption{Una foto de Meteosat, proporcionada por el INM}
\label{fig:meteo}
\end{figure}
```

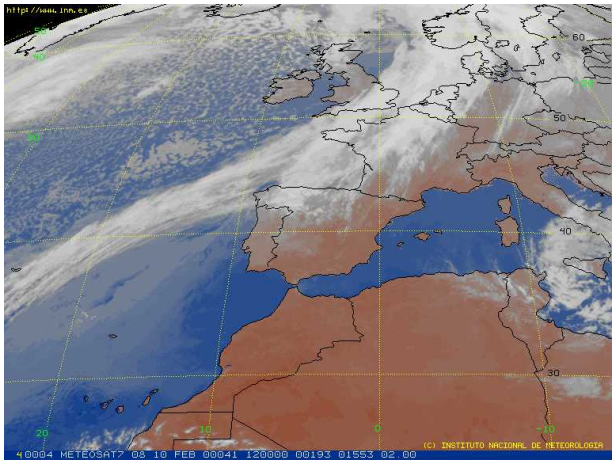
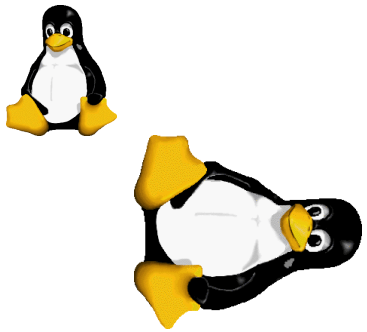


Figura 1: Una foto de Meteosat, proporcionada por el INM

Gráficos para `dvips` y `dvipdfm`

Perrerías al pingüino:



Para `dvips`, convertir todo a EPS. Tiene un tamaño natural, y se pueden cambiar la escala, rotar,...

Para `dvipdfm`, se pueden incluir PDF,JPG,PNG y MetaPost. Como LaTeX no puede leer archivos binarios, no se puede hacer una idea del tamaño natural de los PDF,JPG,PNG. Se incluye una pequeña utilidad que nos ahorra el trabajo:

```
ebb -v mifoto.{pdf,png,jpg}
```

nos crea un archivo `mifoto.bb` con la información del BoundingBox.

¡No hay color!

Volvemos a estar en manos del *DVI-driver*: el soporte al color se define mediante **modelos**: algunas cosas pueden no tener sentido para algún manipulador. Para `dvips` y `dvipdfm`, todo lo que sigue se puede hacer.

Se debe incluir en el preámbulo del documento

```
\usepackage[dvipdfm,dvipsnames,usenames]{color}
```

Puedo definir un color respecto a uno de los siguientes modelos:

`rgb` Contenido de rojo, verde y azul. (aditivo, pantalla).

`cmyk` Contenido de cian, magenta, amarillo y negro (sustractivo, impresora).

`gray` Escala de grises.

`named` 68 colores con nombre propio (además de los ocho nombrados por omisión).

Para definir un color,

```
\definecolor{nombre}{modelo}{nums}
```

o bien en cualquiera de los mandatos siguientes, donde dice `{color}` puedo poner un nombre predefinido o `[modelo]{nums}`.

Para cambiar el color de fondo de la página, de aquí en adelante:

```
\pagecolor[named]{Goldenrod}
```

Declaraciones:

— `\color{color}` Para cambiar el color del texto en adelante.

— `\normalcolor` Para volver al que estaba activo al final del preámbulo.

Mandatos:

```
\textcolor[rgb]{0.3,0.2,0.9}{Vaya texto raro}
```

Vaya texto raro

```
\colorbox[named]{Apricot}{Caja melocotón}
```

Caja melocotón

```
\fcolorbox[named]{Fuchsia}{SpringGreen}{Caja con borde}
```

Caja con borde

De la tiza al proyector pasando por la transparencia

Estilos «clásicos» para presentaciones:

— `slides`, desaconsejado.

— `seminar`, más potente menos actualizado (incluye `fancybox`).

— `foiltex` Sencillo, potente, licencia no del todo libre.

Estilos «futuristas» para presentaciones en PDF: `pdfslide`, `ppower4`.

FoilTEX

Debemos empezar el documento con:

```
\documentclass[opciones]{foils}
```

y eso impone un diseño de página y una elecciones de tipos.

Opciones estándar son: `17pt`,**`20pt`**,`25pt`,`30pt`,`shortform` `headrule`,`footrule`,`landscape`

Se empieza una nueva transparencia con

```
\foilhead[long]{texto}
\rotatefoilhead[long]{texto}
```

Y sólo nos queda por decidir el diseño de los bordes de la transparencia, global o localmente. Por omisión `\MyLogo{ }` y `\Restriction{ }` llenan el ángulo inferior izquierdo, el número de transparencia el ángulo inferior derecho.

Puedo poner lo que quiera en las esquinas superiores con `\leftheader` y `\rightheader`.

El paquete `fancyhdr`, proporciona mucha más flexibilidad; si se usa con foils, debe indicarse en el preámbulo

```
\let\headwidth\textwidth
```

¡Ojo!:

Tiende a «estirar» demasiado las cosas: `\sloppyfoils` o `\enlargethispage*[100cm]`

No hay mandatos de *seccionado* del documento.

Salvo eso, es muy sencillo preparar unas transparencias «decentes».

Hipertexto sin hipo

A partir de unas especificaciones genéricas para todos los *drivers*, S. Rahtz ha escrito el paquete `hyperref`. Está en la versión 6.67e (o más).

Se debe cargar:

```
\usepackage[dvipdfm]{hyperref}
```

Y el resto de las opciones del documento se pueden especificar en el mandato `\hypersetup` que entre otras opciones tiene (para el resto, echar un vistazo a la documentación):

```
\hypersetup{colorlinks,backref,
pdftitle=Mi Tesis,pdfauthor=A. Einstein,pdfsubject=Quantum Chaos,
pdfpagemode=FullScreen}
```

Niveles de hipericia

— Nivel 0: Solo cargar `hyperref`, convierte los `\ref`, `\label`,`\cite`,`\pageref` en hipervínculos. 1

— Nivel 1: Incluir`\href{URL}{lo_que_sea}` o bien `\url{URL}`

— Nivel 2: Posibilidades de HTML, formularios, JavaScript... (léase el manual)

¡Y también puedo hacer páginas web!

LaTeX era un lenguaje de marcado (*mark-up language*) mucho antes de que existiera HTML (HyperText Markup Language, desarrollado en el CERN): parece lógico esperar una «traducción» relativamente fácil.

Evolución: SGML, HTML, XML (MathML, XHTML). En muy poco tiempo, todo será XML.

Lo mejor es la traducción inversa: de SGML, HTML, o XML con un DTD adecuado, sacar un bonito documento TeX.

Opciones: TTH, LaTeX2html, TechExplorer, WebEQ, ...

LaTeX2html tiene un cierto respaldo *oficial*. Consta de unos cuantos programas escritos en Perl; Ha sido desarrollado por Nikos Drakos, Ross Moore y muchos más.

Convierte parte de las ecuaciones en imágenes GIF (Word2000 hace lo mismo).

Gracias a este curso, se ha desentrañado el misterio de su instalación en Win9x/NT, y se puede aprender en
http://feynman.faii.etsii.upm.es/~seidel/l2h/.

Veamos un ejemplo del resultado:

Tesis

Thanks

Este documento reside en:
http://filemon.mecanica.upm.es/~goico/ltxice/miercoles.pdf

y también en:
http://feynman.faii.etsii.upm.es/~seidel/curso2000/sesion3.pdf

y se puede copiar y difundir sin restricciones, salvando los derechos del autor. Se agradecen comentarios, manifestaciones de apoyo, ... aquí.

Nota: En la versión obtenida en la red, es posible que no funcionen adecuadamente algunos enlaces.

Otra nota: Se puede obtener el archivo fuente, para aprender (y comprobar que no hay trucos) aquí.

Ejercicios

1. Estudiar los ejemplos de YAP.

2. Realizar el proceso `.tex` $\rightarrow$ `.pdf` con `sample2e.tex` y con alguno que hayamos escrito.

3. Probar los cambios de tipo, en `sample2e.tex` (también de fórmulas).

4. Visualizar en GSView `tiger.ps` e incluirlo como figura en un documento PS, cambiando opciones.

5. Comprobar el alcance de los cambios de color.

6. Hacer un documento de foiltex, definiendo el diseño de página.

7. Escribir un documento con enlaces, dentro o fuera del propio documento.

8. Comprobar los problemas de las codificaciones T1, OT1 al pasar a PDF.

9. Componer la tesis del autor de WinEDT, cargando el paquete hyperref.

10. **Para casa:** probar LaTeX2html.