

# Usando L<sup>A</sup>T<sub>E</sub>X 1.97

Laura M. Castro Souto (laura@gpul.org)  
Juan José Iglesias González (bille@gpul.org)

16 de marzo de 2004

# Índice general

<b>1. Introducción: Fundamentos de <math>\LaTeX</math></b>	<b>1</b>
1.1. ¿Qué es $\LaTeX$ ?	1
1.2. Características de $\LaTeX$	1
1.3. Lo mejor (o peor) de $\LaTeX$	2
1.3.1. $\LaTeX$ vs. procesadores de texto	2
1.3.2. Y si es tan bueno... ¿por qué no lo usa todo el mundo?	5
1.3.3. Una herramienta útil: KILE	5
1.4. A continuación...	7
<b>2. Las reglas del juego</b>	<b>8</b>
2.1. Fichero $\LaTeX$	8
2.1.1. El preámbulo	8
2.1.2. El contenido	9
2.2. Las ordenes de $\LaTeX$	9
2.2.1. Los caracteres reservados	9
2.2.2. Comandos	10
2.2.3. Entornos	10
2.3. Símbolos especiales	10
2.4. Uso de varios ficheros en un documento	10
2.5. Compilación de un documento $\LaTeX$	10
2.5.1. Compilación clásica	10
<b>3. Diseño básico</b>	<b>13</b>
3.1. Tipos de letra	13
3.1.1. Familias	13
3.1.2. Perfiles	14
3.1.3. Tamaño y grosor	15
3.1.4. Otros efectos	17
3.2. Formato general	17
3.2.1. Separación de palabras y párrafos, interlineado	17
3.2.2. Sangrado y justificación	19

3.2.3.	Segmentacion silábica . . . . .	19
3.3.	Utilidades . . . . .	19
3.3.1.	El euro . . . . .	20
3.3.2.	Citas textuales y versos . . . . .	20
3.3.3.	Un poco de color . . . . .	20
<b>4.</b>	<b>Diseño de documentos</b>	<b>22</b>
4.1.	Clases de documento . . . . .	22
4.2.	Partes del documento . . . . .	22
4.3.	Portadas . . . . .	23
4.4.	Prólogos y secciones especiales . . . . .	24
4.4.1.	Apéndices . . . . .	24
4.4.2.	Casos especiales previstos en book . . . . .	24
4.5.	Índice de contenidos . . . . .	24
<b>5.</b>	<b>Elementos de edición</b>	<b>26</b>
5.1.	Listas . . . . .	26
5.1.1.	Numeradas y no numeradas . . . . .	26
5.1.2.	Descriptivas . . . . .	28
5.2.	Texto en columnas . . . . .	28
5.3.	Notas a pie de página . . . . .	30
<b>6.</b>	<b>Tablas</b>	<b>31</b>
6.1.	Tablas básicas . . . . .	31
6.1.1.	Líneas y separadores . . . . .	32
6.1.2.	Comando extracolsep . . . . .	32
6.1.3.	Multicolumn . . . . .	32
6.1.4.	Rayas horizontales sobre determinadas celdas . . . . .	33
6.2.	Los cuadros . . . . .	33
<b>7.</b>	<b>Referencias, citas bibliográficas e índice de materias</b>	<b>34</b>
7.1.	Referencias . . . . .	34
7.2.	Citas bibliográficas . . . . .	35
7.2.1.	El entorno thebibliography . . . . .	35
7.0.1.	BIB <sub>T</sub> E <sub>X</sub> . . . . .	36
7.1.	Índice de materias . . . . .	39
<b>8.</b>	<b>Imágenes</b>	<b>40</b>
8.1.	El paquete graphicx . . . . .	40
8.2.	El comando includegraphics . . . . .	41
8.2.1.	Opciones de includegraphics . . . . .	41

8.3. Figuras . . . . .	41
<b>9. Fórmulas</b>	<b>43</b>
9.1. Paquetes necesarios . . . . .	43
9.2. El modo matemático . . . . .	43
9.3. Exponentes y subíndices . . . . .	44
9.4. Fracciones y binomios . . . . .	44
9.5. Raíces, integrales, sumatorios, límites . . . . .	45
9.6. Delimitadores . . . . .	46
9.7. Matrices y determinantes . . . . .	46
9.7.1. Puntos suspensivos . . . . .	47
9.8. Sistemas de ecuaciones . . . . .	47
9.9. Teoremas, lemas, corolarios y demostraciones . . . . .	48
9.10. Otros elementos útiles . . . . .	49
9.10.1. Funciones y símbolos . . . . .	49
9.10.2. Texto y fuentes . . . . .	50
9.10.3. Espacios . . . . .	51
9.10.4. Cajas . . . . .	51
<b>10. Más sobre fuentes</b>	<b>53</b>
10.1. La codificación . . . . .	53
10.2. ¿Por qué el pdf se ve mal en Acrobat Reader? . . . . .	54
10.2.1. $\LaTeX$ y su generación de tipos de letra . . . . .	54
10.2.2. Usar fuentes PostScript . . . . .	55
<b>11. <math>\LaTeX</math> y el hipertexto</b>	<b>57</b>
11.1. Exportar a HTML . . . . .	57
11.1.1. tex4ht . . . . .	57
11.1.2. latex2html . . . . .	57
<b>12. Presentaciones</b>	<b>58</b>
12.1. Esquema de la clase prosper . . . . .	58
12.2. Estilos de la presentación . . . . .	59
12.3. Más opciones de la clase prosper . . . . .	59
12.4. Como hacer las transparencias . . . . .	60
12.4.1. Otros comandos . . . . .	60

## **Resumen**

Documentos de tipo técnico o científico demuestran la incapacidad de los procesadores de textos para resolver aspectos claves de diseño (índices, tablas, ecuaciones, bibliografía...).  $\text{\LaTeX}$  permite la composición de textos de forma semiautomática, con una gran calidad, y se puede exportar a formatos abiertos como HTML, PS o PDF.

# Capítulo 1

## Introducción: Fundamentos de L<sup>A</sup>T<sub>E</sub>X

### 1.1. ¿Qué es L<sup>A</sup>T<sub>E</sub>X?

L<sup>A</sup>T<sub>E</sub>X<sup>1</sup> es un **sistema de composición de textos** orientado al ámbito científico-técnico (libros, artículos, cartas, ...).

### 1.2. Características de L<sup>A</sup>T<sub>E</sub>X

- Un fichero L<sup>A</sup>T<sub>E</sub>X se compone de texto acompañado de una serie de comandos que definen tanto el significado como el formato de cada parte del texto. Podría considerarse similar al caso de HTML, por ejemplo, en el sentido de que el texto va acompañado de *marcas*; sin embargo, la diferencia estriba en que L<sup>A</sup>T<sub>E</sub>X está pensado para ser impreso.
- L<sup>A</sup>T<sub>E</sub>X da al texto escrito una estructura y formato predefinidos que permite una mayor legibilidad, ya que se usan estilos de composición de texto utilizados durante años en el ámbito de la maquetación e impresión.
- Todos los elementos de edición comunes (tablas, imágenes, listas, índices de todo tipo —de materias, de figuras, de tablas, glosario, etc.—) son soportados por L<sup>A</sup>T<sub>E</sub>X.

---

<sup>1</sup>Para los detallistas, pronunciado “*lei-tegh*”.

- $\LaTeX$  es un estándar abierto, lo que permite su exportación a otros formatos, como HTML, PS o PDF. Además, podemos usar  $\LaTeX$  en prácticamente cualquier plataforma.
- $\LaTeX$  es muy adecuado para el tratamiento de expresiones matemáticas.
- $\LaTeX$  cuenta con una gran comunidad de usuarios. Gracias a ello, disponemos de mucha ayuda y recursos *online*, y también de una gran cantidad extensiones para campos concretos (que van desde la música o la química molecular hasta los circuitos eléctricos o el ajedrez), o que aportan nuevas funcionalidades (documentos con un número variable de columnas, transparencias, colores...).
- $\LaTeX$  soporta múltiples lenguajes.

### 1.3. Lo mejor (o peor) de $\LaTeX$

Muchos usuarios consideran que es engorroso trabajar con un fichero de texto y tener que aprender una serie de comandos a insertar entre el propio texto para indicar su estructura y formato. Consideran que esta manera de trabajar es primitiva.

Sin embargo, la gente que conoce  $\LaTeX$  se da cuenta que trabajar de esta forma permite, en proyectos de cierta consideración, ahorrar tiempo y esfuerzo, pues al contrario de lo que ocurre con la gran mayoría de los procesadores de texto, los aspectos relacionados con composición y estética se automatizan en gran parte. Además, el resultado obtenido es siempre de calidad y usualmente tiene un aspecto mucho más cuidado y profesional.

#### 1.3.1. $\LaTeX$ vs. procesadores de texto

La mayoría de herramientas más usadas para escribir documentos emplean una filosofía distinta a la de  $\LaTeX$ , conocida como WYSIWYG<sup>2</sup>. Los procesadores de texto WYSIWYG muestran durante la propia edición una representación gráfica del documento que pretende ser un reflejo de la versión impresa que producirá. Sin embargo, a la hora de la verdad, lo que se ve es lo que el editor estima oportuno o, peor aún, lo que se imprime no coincide con lo que el editor muestra.

---

<sup>2</sup>*“What you see is what you get”, es decir, “lo que ves es lo que obtienes”.*

El manejo moderado o profuso de marcos, ecuaciones, listas e índices en procesadores de texto de este tipo suelen provocar la desesperación de los sufridos usuarios.

Los principales problemas de los procesadores WYSIWYG son:

- El autor pierde tiempo ocupándose de todas las cuestiones referentes al diseño del documento.
- El usuario de procesador de textos no tiene por qué conocer las reglas de diseño de documentos. Por ello, es normal que no tome las decisiones adecuadas<sup>3</sup>. Cuando tenemos entre manos un documento de varias páginas ésta no tiene por qué ser una cuestión trascendental, pero si hablamos de un libro o un proyecto científico, entonces claramente sí lo es.
- Los actuales procesadores de texto desarrollan más la parte visual o decorativa<sup>4</sup> que las propias herramientas de composición de textos. Las áreas olvidadas o incómodas de usar en procesadores WYSIWYG siguen siendo las mismas que hace 10 años: espaciado de palabras, colocación de objetos flotantes (tablas y figuras), mala gestión de referencias e índices (temarios, glosarios, lista de tablas, bibliografías, etc) y otras muchas.
- Muchos procesadores de texto utilizan formatos cerrados que impiden la compartición de documentos entre usuarios y plataformas<sup>5</sup>.

¿Cuál es la contrapartida que ofrece L<sup>A</sup>T<sub>E</sub>X ante este panorama? L<sup>A</sup>T<sub>E</sub>X usa T<sub>E</sub>X, un gran programa que garantiza una elevada calidad y precisión en la composición de textos. El desarrollador de T<sub>E</sub>X<sup>6</sup> fue Donald Knuth, uno de los grandes nombres de la historia de la Informática. Al recibir la prueba de su libro "*The Art of Programming*", quedó tan descontento que emprendió la tarea de desarrollar un sistema que le permitiese editar su propio libro.

Más tarde, Leslie Lamport desarrolló L<sup>A</sup>T<sub>E</sub>X, para que los usuarios no tuvieran que tratar directamente con T<sub>E</sub>X, que si bien era un lenguaje muy potente y efectivo, también era excesivamente complejo. Definiendo una serie de macros y plantillas, L<sup>A</sup>T<sub>E</sub>X permite producir documentos bien diseñados donde la legibilidad es el principal objetivo (tipos de letra adecuados, numeración estandarizada, longitud de los renglones que no fatiguen

---

<sup>3</sup>Error típico: usar pocos tipos de letra o, por el contrario, usar demasiados. Ambos son incómodos para el lector.

<sup>4</sup>Por ejemplo, Word y su pinball incorporado.

<sup>5</sup>Algunos editores son incluso incompatibles entre sus propias versiones.

<sup>6</sup>Para los detallistas, pronunciado "*tegh*".

al lector, justificación de líneas y párrafos, etc.). La estructura de un documento de este tipo es fácilmente reconocible y obedece a los requisitos que debe presentar un documento técnico o científico.

### 1.3.2. Y si es tan bueno... ¿por qué no lo usa todo el mundo?

- Empezar a usar  $\LaTeX$ , incluso de un modo básico, exige aprender una serie de comandos.
- La gente está más acostumbrada a los procesadores WYSIWYG y suele desconocer  $\LaTeX$ .
- $\LaTeX$  toma muchas decisiones de manera automática, casi siempre acierta, pero a veces no. En ocasiones corregir sus errores no es trivial<sup>7</sup>.
- El proceso de crear o modificar plantillas o características de  $\LaTeX$  puede ser algo más complicado.
- $\LaTeX$  está poco orientado a gráficos y color, hay que recurrir a paquetes suplementarios (y, por tanto, aprender nuevos comandos) para poder usar cómodamente este tipo de elementos.

### 1.3.3. Una herramienta útil: KILE

Kile es un editor de textos desarrollado por P. Brachet. Está basado en *kate* y por lo tanto integrado en **KDE**. Tiene una completa interfaz con diversas facilidades que nos permitirán subsanar los principales “peros” de un usuario novel:

- Los comandos de  $\LaTeX$  están disponibles a través de menús, botones y combinaciones de teclas.
- La ayuda integrada en el programa nos permitirá saber qué macro usar ante una necesidad concreta.
- Para una edición cómoda de los ficheros de texto, contamos con resaltado de sintaxis, funciones de búsqueda (incremental o no), reemplazo, deshacer, corrección ortográfica...

---

<sup>7</sup>Ejemplo típico: por defecto introduce un pequeño espacio tras un punto, por considerarlo fin de frase. Ahora bien, cuando escribimos un acrónimo como O.N.U., no deseáramos que eso ocurriera.

- Los más de 370 símbolos matemáticos posibles son accesibles asimismo mediante botones y menús.
- Asistentes para la creación de distintos tipos de documentos  $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$  (cartas, artículos, etc).
- Manejo de bibliografías a través de  $\text{BIB}_{\text{T}}\text{E}_{\text{X}}$ .
- Navegación mediante menús de la estructura de un documento o proyecto.
- Facilidades para compilar y depurar ficheros  $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ .
- Integración con herramientas externas para la visualización e impresión de los documentos editados en distintos formatos: DVI, POSTSCRIPT o PDF.
- Interfaz con programas de dibujo como xfig o gnuplot.

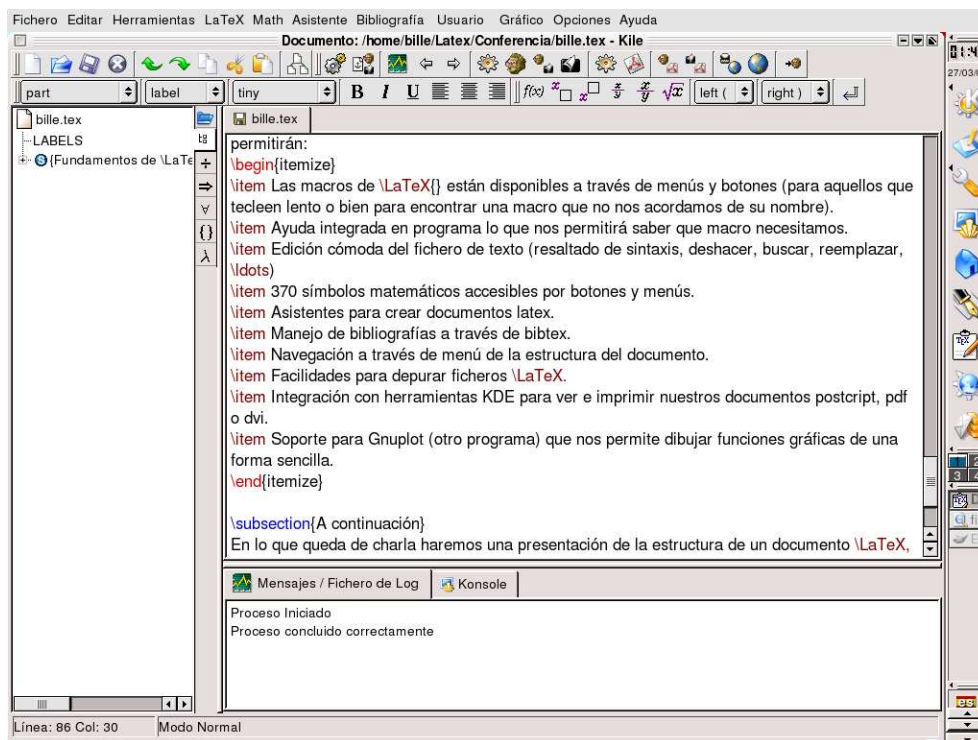


Figura 1.1: Una captura de kile

## 1.4. A continuación...

Una vez introducido, nos adentraremos ahora en el mundo de  $\text{\LaTeX}$ . Presentaremos la estructura básica de un documento  $\text{\LaTeX}$ , y las tareas de compilación y depuración. Veremos las principales macros y cómo deben ser usadas para incluir los elementos típicos: formatos de letra, listas, enumeraciones, notas, tablas, citas, índices, imágenes, figuras, referencias...

# Capítulo 2

## Las reglas del juego

### 2.1. Fichero $\text{\LaTeX}$

Un documento  $\text{\LaTeX}$  se divide en dos partes fundamentales: el preámbulo y el contenido del documento:

#### 2.1.1. El preámbulo

Se define como la parte del fichero que va antes del contenido. Se usa para incluir información sobre la naturaleza del documento, los comandos que definimos nosotros mismos y para incluir los paquetes opcionales que queramos usar en nuestro diseño de documentos.

```
\documentclass[a4paper,12pt]{article}
\usepackage[latin1]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[spanish]{babel}
\title{Título del texto}
\author{Autor del texto}
```

En este ejemplo hacemos lo siguiente:

1. Se define el tipo de documento (artículo, tipo de letra base de 12 puntos y hojas a4).
2. Se incorpora el paquete `inputenc` para que acepte símbolos como letras con tilde, eñes, ...
3. Se incorpora el paquete `fontenc` para que se codifiquen los símbolos especiales con los códigos correspondientes y no con combinaciones de caracteres.

4. Se carga el paquete babel que modifica el comportamiento de  $\text{\LaTeX}$  para adaptarlo a idiomas distintos del inglés.
5. Se define el título del documento.
6. Se define el autor del documento.

### 2.1.2. El contenido

El resto del documento (el contenido junto con las diferentes ordenes de diseño) se introducirá dentro del entorno document, en el ejemplo incluimos la orden maketitle para que  $\text{\LaTeX}$  construya una portada de nuestro documento.

```
\begin{document}
\maketitle
Texto a continuación...
\end{document}
```

## 2.2. Las ordenes de $\text{\LaTeX}$

En los ejemplos anteriores hemos empezado a ver diferentes tipos de ordenes de  $\text{\LaTeX}$ , podemos agruparlos en caracteres reservados, comandos y entornos.

### 2.2.1. Los caracteres reservados

$\text{\LaTeX}$  reserva 10 caracteres para la sintaxis de sus órdenes. Cada vez que los usemos el sistema entenderá que queremos dar algún tipo de orden. Si los queremos usar en nuestro contenido deberemos usar comandos especiales que muestran esos símbolos. Los caracteres reservados son los siguientes:

$\backslash$     $\{$     $\}$     $\#$     $\&$     $\%$     $\sim$     $\_$     $\^$     $\$$

- La barra invertida es el comienzo de una orden de  $\text{\LaTeX}$ .
- Las llaves se utilizan para agrupar bloques de código (como un lenguaje de programación).
- $\#$  Se utiliza para nombrar los diferentes argumentos de un comando que queramos definir.

- `&` Se utiliza para separar las columnas de una tabla o matriz.
- `%` Se utiliza para marcar una línea como comentario ( $\text{\LaTeX}$  no tratará lo que venga a continuación).
- `_ ^ $` Se utilizan en fórmulas matemáticas (subíndice, superíndice, marcar comienzo y fin de una fórmula).

### 2.2.2. Comandos

$\text{\LaTeX}$  y los diferentes paquetes que se han construido sobre él definen una gran cantidad de comandos con las más diversas funciones. Su forma general es la siguiente:

```
\nombredecomando[argumento_opcional]{argumento obligatorio}
```

Muchas veces tendremos que encerrar el comando entre llaves, lo que hará que el efecto del comando se restrinja al bloque que encierra las llaves.

### 2.2.3. Entornos

Los entornos en cambio son comandos que claramente definen un bloque por si mismos ya que tienen etiqueta de cierre, por lo que su efecto será restringido al bloque que encierran .

```
\begin[argumento_opcional]{argumento_obligatorio}{nombre_de_entorno}
... contenido dependiente del tipo de entorno ...
\end{nombre_de_entorno}
```

## 2.3. Símbolos especiales

## 2.4. Uso de varios ficheros en un documento

## 2.5. Compilación de un documento $\text{\LaTeX}$

### 2.5.1. Compilación clásica

El modo tradicional de trabajar con  $\text{\LaTeX}$  era mediante un editor de texto (`vi`, `emacs`, ...) con lo que conseguimos un fichero `.tex` que después se compilaría al formato DVI. La orden a usar sería la siguiente:

Símbolo	Teclado	Resultado
Comillas españolas	<< >>	« »
Comillas americanas	\ \ ' '	“ ”
Comillas inglesas	\ ' '	‘ ’
Guión corto <sup>1</sup>	-	-
Guión medio <sup>2</sup>	--	-
Raya <sup>3</sup>	---	—
Puntos suspensivos <sup>4</sup>	...	...
Ordinales	1\textsuperscript{er}	1 <sup>er</sup>
Ordinales <sup>5</sup>	1\sptext{er}	1. <sup>er</sup>
Grados	25\textdegree	25°

Cuadro 2.1: Símbolos especiales

Signo	\	{	}	#	&	%	~	\$	-	^
Comando	\texbackslash	\{	\}	\#	\&	\%	\~{}	\\$	\_	\^{}

Cuadro 2.2: Escapes para los símbolos L<sup>A</sup>T<sub>E</sub>X

```
%latex nombrededefichero.tex
```

De esta manera iniciaremos una compilación «interactiva» donde el sistema nos irá dando los errores y advertencias correspondientes. Ante un error el usuario puede:

- Pulsar ‘x’ para parar la compilación.
- Pulsar ‘e’ para editar (se abre el \$EDITOR).
- Pulsar ‘enter’ para que intente compilar hasta donde pueda.
- Pulsar ‘s’ y despues ‘ctrl+c’ para salir (si falla lo anterior).
- Pulsar ‘q’ para entrar en «non-stop mode»

### Non-Stop Mode

El compilador empezará a tratar el fichero correspondiente guardando en NOMBREDEFICHERO.LOG información acerca del proceso (incluyendo errores y avisos). Este es el modo que usa KILE.

## **Tipos de errores**

Como en todo lenguaje de programación podremos tener errores de sintaxis al introducir mal las órdenes, colocar mal algún argumento o simplemente olvidarnos de alguna llave.

Entre el coloreado de sintaxis que suelen usar los editores de texto (como KILE o emacs) y los errores al compilar deberemos ser capaces de corregirlos aunque como todo la depuración del fichero dependen mucho de nuestra práctica.

KILE incorpora además ayudas con los bloques (iluminación de los caracteres de apertura y cierre de bloque) así como a la hora del auto-completado de código.

# Capítulo 3

## Diseño básico

Trataremos a continuación con los principales elementos de formato de un documento, como son los tipos de letra, la indentación y alineado de texto, etc. Después, veremos también algunas curiosidades, como el uso del color o la inclusión del símbolo del euro.

### 3.1. Tipos de letra

En primer lugar, debe recordarse que  $\text{\LaTeX}$  es un sistema de composición de textos que pretende, entre otras cosas, evitar al usuario gran parte de las cuestiones estéticas de diseño del documento, haciendo que sólo tenga que especificar sus líneas maestras.

Por supuesto,  $\text{\LaTeX}$  incorpora además la posibilidad de personalizar y modificar el formato estándar que da al texto utilizando distintos comandos y macros, que veremos a continuación. Sin embargo, aunque pueden usarse en casos concretos, no es recomendable abusar de estas características, pues es probable que caigamos en defectos de diseño que repercutan en la generación de un documento poco agradable a la vista del lector, rompiendo con el aspecto cuidado y armonioso que  $\text{\LaTeX}$  proporciona.

#### 3.1.1. Familias

$\text{\LaTeX}$  provee de tres familias de tipos de letra:

**Roman** (por defecto)

que se indica mediante el comando `\textrm{texto}`  
y produce “este tipo de letra”.

**Sanserif** (o sin adornos)

que se indica mediante el comando `\textsf{texto}`  
y produce “este tipo de letra”.

**Máquina de escribir** (o typewriter)

que se indica mediante el comando `\texttt{texto}`  
y produce “este tipo de letra”.

La familia Roman, que es la que L<sup>A</sup>T<sub>E</sub>X utiliza por defecto, no es necesario indicarla, pero puede hacerse para recuperar el tipo habitual dentro de un contexto donde esté activa otra familia. Por ejemplo, el código:

```
\texttt{Aquí está activa la familia
      typewriter \textrm{pero puedo recuperar
                  la familia Roman}
      si lo deseo}.
```

Produce la siguiente salida:

```
Aquí está activa la familia typewriter pero pue-
do recuperar la familia Roman si lo deseo.
```

Como se puede observar, la inclusión de múltiples espacios y tabuladores es ignorada por L<sup>A</sup>T<sub>E</sub>X, que los interpreta como un espacio simple.

### 3.1.2. Perfiles

Cada una de las familias de tipos de letra que proporciona L<sup>A</sup>T<sub>E</sub>X por defecto tiene cuatro perfiles diferentes:

**Recto** (por defecto)

que se indica mediante el comando `\textup{texto}`  
y produce “este tipo de letra”.

**Itálico**

que se indica mediante el comando `\textit{texto}`  
y produce “*este tipo de letra*”.

**Inclinado**

que se indica mediante el comando `\textsl{texto}`  
y produce “*este tipo de letra*”.

**Versalita**

que se indica mediante el comando `\textsc{texto}`  
y produce “ESTE TIPO DE LETRA”.

Como ya hemos comentado, los comandos pueden combinarse, anidándolos, de la forma en que se desee:

```
\textsf{Este tipo es sanserif
      \textsl{inclinado}
      y también \textsc{en versalita}
    }.
```

Este tipo es sanserif *inclinado* y también EN VERSALITA.

### 3.1.3. Tamaño y grosor

Aunque, como vemos, L<sup>A</sup>T<sub>E</sub>X ya se encarga de resaltar los elementos destacables de un documento aumentando el tamaño de letra y/o poniendo en negrilla cosas como títulos de capítulos y secciones, por ejemplo, puede ser necesario para nosotros poder aplicar esas modificaciones directamente.

Así, disponemos de dos grosores de letra:

**Normal** (o medio)

que se indica mediante el comando `\textmd{texto}`  
y produce “este tipo de letra”.

**Grueso** (o negrita)

que se indica mediante el comando `\textbf{texto}`  
y produce “**este tipo de letra**”.

Y de 10 variaciones de tamaño de letra:

**Huge**

que se indica mediante el comando `{\Huge texto}`  
y produce “**este tipo de letra**”.

**huge**

que se indica mediante el comando `{\huge texto}`  
y produce “**este tipo de letra**”.

**LARGE**

que se indica mediante el comando `{\LARGE texto}`  
y produce “**este tipo de letra**”.

**Large**

que se indica mediante el comando `{\Large texto}`  
y produce “este tipo de letra”.

**large**

que se indica mediante el comando `{\large texto}`  
y produce “este tipo de letra”.

**normalsize**

que se indica mediante el comando `{\normalsize texto}`  
y produce “este tipo de letra”.

**small**

que se indica mediante el comando `{\small texto}`  
y produce “este tipo de letra”.

**footnotesize**

que se indica mediante el comando `{\footnotesize texto}`  
y produce “este tipo de letra”.

**scriptsize**

que se indica mediante el comando `{\scriptsize texto}`  
y produce “este tipo de letra”.

**tiny**

que se indica mediante el comando `{\tiny texto}`  
y produce “este tipo de letra”.

Es importante notar que estas variaciones en el tamaño de la letra siempre serán proporcionales al tamaño base que se haya indicado en las opciones del comando `documentclass`.

En ocasiones, según la longitud del texto cuyas características queramos alterar, puede ser más recomendable utilizar un *entorno* en lugar de un comando. Para cada uno de los comandos que acabamos de ver, el entorno equivalente se construye de distintas maneras según la característica:

Familia de tipo de letra	<code>textZZ</code>	⇒	Entorno	<code>ZZfamily</code>
Perfil de tipo de letra	<code>textYY</code>	⇒	Entorno	<code>YYshape</code>
Grosor de letra	<code>textXX</code>	⇒	Entorno	<code>XXseries</code>

En el caso de los tamaños de letra, los entornos tienen los mismos nombres. Veamos un ejemplo:

```

\begin{sffamily}
Este es un bloque de letra sanserif
\begin{Large}
con entorno de texto mayor en medio
\end{Large}
y también
\begin{slshape}
un entorno inclinado.
\end{slshape}
\end{sffamily}

```

Produce:

Este es un bloque de letra sanserif **con entorno de texto mayor en medio** y también *un entorno inclinado*.

### 3.1.4. Otros efectos

Otra manera de enfatizar un texto en medio de otro es utilizar el comando `\emph{texto}`. Si el entorno en que utilizamos este comando es de texto normal, el *texto* insertado aparecerá en itálica, mientras que si el entorno es itálico, aparecerá recto.

Con el comando `\underline{texto}` se puede subrayar una selección con este resultado.

Por último, habíamos comentado que  $\LaTeX$  ignora los espacios, tabuladores e incluso líneas en blanco que podamos añadir de más por error. Si en alguna circunstancia nos interesa que el texto que introducimos aparezca tal y como lo tecleamos, es decir, respetando espacios, saltos de línea y caracteres reservados, podemos utilizar el comando `\verb[?]texto[?]`, donde `[?]` es un carácter de nuestra elección con el que indicamos el comienzo y el final de la secuencia que ha de ser respetada por  $\LaTeX$ . El entorno equivalente se denomina *verbatim*. En estos casos,  $\LaTeX$  emplea siempre la familia `typewriter`.

## 3.2. Formato general

### 3.2.1. Separación de palabras y párrafos, interlineado

Un salto de línea simple no producirá en  $\LaTeX$  el efecto esperado por un usuario acostumbrado a los procesadores WYSIWYG. Para conseguir un

cambio de línea o de párrafo, es necesario usar alguna de las siguientes cadenas (totalmente equivalentes):

```
\newline  
\\  
\par
```

Análogamente, para forzar a que se produzca un cambio de página, disponemos de los comandos:

```
\newpage  
\clearpage  
\cleardoublepage
```

La diferencia entre ellos radica en su comportamiento ante la presencia de objetos “flotantes” (figuras, tablas, etc) que aún no hayan sido colocados a la llegada del comando; `\clearpage` y `\cleardoublepage` los ubicarán en páginas sin texto, siendo el segundo la versión que debe usarse si se está creando un documento para el que se ha especificado la opción `twoside`.

También se pueden insertar artificialmente espacios en horizontal, con el comando `\hspace{longitud}`, o en vertical, con `\vspace{longitud}` (o incluso con `\[longitud]`), aunque para variar la separación vertical entre líneas quizás sea más útil redefinir la variable que contiene el valor de dicha separación:

```
\renewcommand{\baselinestretch}{longitud}
```

Claro que si el tratamiento ha de aplicarse al documento entero (por ejemplo, para escribir una carta a doble espacio), es más sencillo utilizar el paquete `setspace` (`\usepackage{setspace}`), que nos provee de los intuitivos comandos:

```
\singlespacing  
\onehalfspacing  
\doublespacing
```

### 3.2.2. Sangrado y justificación

Como podemos observar, sin ir más lejos, en el presente documento,  $\LaTeX$  introduce por defecto un sangrado en la primera línea de cada párrafo. Este es el comportamiento adecuado en la presentación de la gran mayoría de documentos, pero si queremos evitarlo podemos utilizar `\noindent` al principio del párrafo.

Éste es el efecto que se consigue.

En cuanto a la alineación del texto, también podemos ver que  $\LaTeX$  lo justifica a ambos lados por defecto, pero podemos:

alinear el texto en medio, utilizando el entorno *center*

alinearlo a la derecha, utilizando el entorno *flushright*

o bien alinearlo sólo a la izquierda, utilizando el entorno *flushleft*.

### 3.2.3. Segmentación silábica

Si en el preámbulo utilizamos la opción del paquete `babel` adecuada al idioma que emplearemos a la hora de redactar el documento (`spanish`, por ejemplo, si pensamos redactar en castellano),  $\LaTeX$  sabrá partir correctamente las palabras al final de una línea cuando sea necesario, siguiendo las normas generales de segmentación silábica del idioma correspondiente.

No obstante, en el caso de palabras que representen tecnicismos, extranjerismos o similares, el resultado puede no ser el apropiado o el que esperamos. En estos casos, hay dos maneras esenciales de actuar:

1. Indicando a  $\LaTeX$  explícitamente por dónde puede romper la palabra. Esto se hace insertando `\-` en dichos puntos de ruptura, por ejemplo, para la palabra *atributo*, indicaríamos `atri\bu\to`.
2. Incluyendo la palabra en una lista de `hyphenation`:

```
\hyphenation{va-ria-ble,me-to-do}
```

## 3.3. Utilidades

Para cerrar este capítulo, comentaremos algunas cuestiones que pueden sernos de utilidad.

### 3.3.1. El euro

L<sup>A</sup>T<sub>E</sub>X es anterior a la entrada en actividad del euro, e incluso a su propia concepción, pero eso no impide que se haya desarrollado un paquete, denominado `eurosym`, cuya inclusión nos permite utilizar el comando `\euro` para obtener este resultado: €.

### 3.3.2. Citas textuales y versos

Para la inclusión de citas y versos en nuestros documentos, L<sup>A</sup>T<sub>E</sub>X dispone de entornos especiales, `quote` y `quotation`, y `verse`. El efecto que producen es el que podemos observar:

El texto de una cita con `quote` se sangra por ambos lados un poco más que el texto normal, de manera que se resalta en medio de éste, tal y como se pretende.

El texto de una cita con `quotation` se sangra por ambos lados un poco más que el texto normal, de manera que se resalta en medio de éste, pero además respeta la indentación.

Dices que tienes corazón y sólo  
lo dices por que sientes sus latidos.  
Eso no es corazón. . . ; es una máquina  
que al compás que se mueve hace ruido.

### 3.3.3. Un poco de color

Para poner un poco de color en la vida de nuestros documentos L<sup>A</sup>T<sub>E</sub>X está a nuestra disposición el paquete `color`, que nos permitirá utilizar comandos como `\textcolor{nombrecolor}{texto}` para colorear de distintas maneras nuestro texto. Además de los `nombrecolor` por defecto (`black`, `white`, `red`, `green`, `blue`, `cyan`, `magenta` y `yellow`), podemos emplear el comando:

```
\definecolor{nuevonombrecolor}{modelo}{especificación}
```

donde `modelo` = [RGB, HSB, CMYK, Gray o Named] y `especificación` depende del modelo.

Otros comandos pertenecientes a este paquete son:

- `\pagecolor{nombrecolor}`, que cambia el color de fondo de las páginas.
- `\colorbox{nombrecolor}{objeto}`, que crea una caja con el fondo del color indicado y en su interior el *objeto* L<sup>A</sup>T<sub>E</sub>X indicado: así.
- `\fcolorbox{nombrecolor}{nombrecolor2}{objeto}`, que crea una caja con color de fondo *nombrecolor2* y un marco de color *nombrecolor* y coloca en su interior el *objeto* indicado: así.

# Capítulo 4

## Diseño de documentos

### 4.1. Clases de documento

La clase del documento es lo que define los parámetros básicos de diseño y estructura nuestra obra.  $\text{\LaTeX}$  en principio maneja los siguientes tipos: `book`, `report` (un informe que es un libro simplificado), `article`, `proc` (actas, variación de `article`), `letter`, `slides` (en desuso). Fuera de los oficiales nos encontramos con algunos como `seminar` (transparencias para retroproyector), `examdoc` (examen), `label` (etiquetas para pegar) y muchos más<sup>1</sup>.

Vamos a tomar como ejemplo la clase `book`, usaríamos:

```
\documentclass[opcion, opcion]{book}
```

Esta orden casi siempre es la primera en el preámbulo. Según el tipo de documento podremos modificar ciertas opciones:

- Tamaño del papel (`a4`, `a3`, `letterpaper`).
- Dos columnas (`twocolumn`).
- Tamaño base de la letra base (`10pt`, `11pt`, `12pt`).
- Galerada <sup>2</sup> (`draft`).

### 4.2. Partes del documento

$\text{\LaTeX}$  según la clase de documento que usemos nos propondrá que dividamos el contenido en diferentes partes.

---

<sup>1</sup>Ver documentación de `tetex`

<sup>2</sup>Versión de prueba para corregir antes de llevar a imprenta

Nombre	Clase article	Clase book
Parte	<code>\part (optativa)</code>	<code>\part (optativa)</code>
Capítulo	no permitido	<code>\chapter</code>
Sección	<code>\section</code>	<code>\section</code>
Subsección	<code>\subsection</code>	<code>\subsection</code>
Subsubsección	<code>\subsubsection</code>	<code>\subsubsection</code>
Párrafo	<code>\paragraph</code>	<code>\paragraph</code>
Subpárrafo	<code>\subparagraph</code>	<code>\subparagraph</code>

Cuadro 4.1: Partes de un documento

Cada una de esos comandos recibe como argumento principal el nombre completo, argumento opcional un nombre breve que se incluirá en el índice y puede tener un \* que servirá para indicar que esa parte no se incluirá en el índice.

```
\chapter*[nombrecorto]{nombre largo}
```

### 4.3. Portadas

Con los siguientes comandos que se incluyen en el preámbulo podremos modificar la información de la portada del documento, hacer una portada básica e incluir un resumen del contenido:

```
\title{Título del Documento \\ Otra línea del título del documento}
\author{ Nombre del Autor1 \and Autor2}
\date{FechaTexto} % si no se pone es como si se pusiera \today
\thanks{Texto} % Se puede poner dentro del author o title y es una es
                % de nota al pie de la portada
\begin{document}
\maketitle
\begin{abstract}
Resumen del contenido del documento (muy recomendable en clase article)
\end{abstract}
\end{document}
```

La portada que crea L<sup>A</sup>T<sub>E</sub>X es muy básica. En un libro profesional se suelen hacer páginas especiales con diseño gráfico y alguna imagen atractiva que sirva de reclamo. Para estas páginas que forman la portada y la contraportada de un libro que se va a encuadenar L<sup>A</sup>T<sub>E</sub>X proporciona el entorno

`titlepage` que lo único que hace es crear una página que no va a numerar y en la que podremos definir los márgenes que queramos sin alterar el resto del documento.

## 4.4. Prólogos y secciones especiales

Para incluir páginas especiales como agradecimientos o licencia del documento podemos usar el entorno `titlepage`. Si queremos un prólogo lo más sencillo puede ser incluir el siguiente capítulo al principio:

```
\chapter*{Prólogo}
```

### 4.4.1. Apéndices

Con `\appendix` creamos los apéndices que serán renumerados con letras mayúsculas. Según la clase que usemos un apéndice será considerado un capítulo (`book`) o bien una sección (`article`).

### 4.4.2. Casos especiales previstos en `book`

La clase `libro` nos permite además:

```
\begin{document}
\frontmatter
% las páginas serán numeradas en números romanos
% los capítulos definidos no sacarán epígrafe "Capítulo número"
% otras partes del documento se tratarán igual (mejor ponerles en ast
\mainmatter
% aquí se incluirán los capítulos que constituyen el libro
% se renumera desde 1
\backmatter
% los apéndices
\end{document}
```

## 4.5. Índice de contenidos

Una de las mayores ventajas de  $\text{\LaTeX}$  es que la creación de índices es totalmente automática. El usuario no tiene que andar revisando los números de página de cada capítulo, el sistema lo hará por él.

```
\begin{document}
\maketitle
\tableofcontents % genera el índice general
                 % habrá que compilar el fichero dos veces para que s
\end{document}
```

# Capítulo 5

## Elementos de edición

Hasta ahora hemos visto elementos esenciales y avanzados de estructuración y formateo de documentos. Completaremos ahora un poco más nuestra visión, con el repaso de algunas funcionalidades muy útiles.

### 5.1. Listas

#### 5.1.1. Numeradas y no numeradas

El principal tipo de listas de ítems en un documento son las *listas numeradas*, del estilo:

1. Un triste tigre
2. Dos tristes tigres
3. Tres tristes tigres

que se construyen utilizando el entorno *enumerate*, y las *listas no numeradas*, simples relaciones de elementos:

- Hacer la compra
- Ir al gimnasio
- Sacar la basura

que se construyen utilizando el entorno *itemize*. Por supuesto, las listas pueden anidarse y combinarse entre sí (éstas y los demás tipos, claro):

- Hacer la compra

- Comprar leche
- Comprar cereales
- Ir al gimnasio
  1. 15 minutos de bicicleta estática
  2. 30 minutos de máquinas
    - 5 minutos de abdominales
    - 10 minutos de piernas
    - 10 minutos de brazos
    - 5 minutos de espalda
  3. 5 minutos de sauna
- Sacar la basura

Como alternativa al entorno *itemize*, L<sup>A</sup>T<sub>E</sub>X nos brinda el entorno *list*, donde es el usuario el que personaliza la etiqueta de la lista:

```
\begin{list}{\clubsuit}{}
  \item Hacer la compra
  \item Ir al gimnasio
  \item Sacar la basura
\end{list}
```

- ♣ Hacer la compra
- ♣ Ir al gimnasio
- ♣ Sacar la basura

Además de estas formas básicas, existe la posibilidad de utilizar el paquete *enumerate*, que nos permite suministrar un parámetro opcional, un patrón de cómo ha de ser la enumeración. Por ejemplo:

```
\begin{enumerate}[1.-]
  \item Un tigre
    \begin{enumerate}[a:=]
      \item Un triste tigre
      ...
    \end{enumerate}
  ...
\end{enumerate}
```

resultaría en

- 1.- Un tigre
  - a:= Un triste tigre
  - b:= ...
- 2.- ...

### 5.1.2. Descriptivas

Además de las listas numeradas y no numeradas, disponemos de un tipo especial de lista en el que el ítem resaltado es un nombre o concepto que se indica. Así:

```
\begin{description}
\item [Objeto] Entidad compleja provista de
           datos y comportamiento
\item [Clase] Conjunto de objetos que comparten
           propiedades y comportamiento
\item [Herencia] Mecanismo que jerarquiza las
           clases de objetos en un sistema
\item [...]
\end{description}
```

genera como salida

**Objeto** Entidad compleja provista de datos y comportamiento

**Clase** Conjunto de objetos que comparten propiedades y comportamiento

**Herencia** Mecanismo que jerarquiza las clases de objetos en un sistema

...

## 5.2. Texto en columnas

Entre las opciones del comando `documentclass`, en el preámbulo del documento, podemos indicar `twocolumn`, de manera que todo el texto que componga el mismo será dispuesto en forma de columnas periodísticas.

No obstante, si tan sólo colocar en forma de columna un trozo de texto, podemos emplear el comando `\twocolumn[ cabecera ]` a partir del momento en que queramos hacerlo.

Esto provocará que el texto se disponga nuevamente en dos columnas periodísticas, tras un salto de página. Si se indica una *cabecera* opcional, ésta presidirá la página. Para volver al modo de edición normal se utiliza el comando `\onecolumn`.

Como se puede ver, estos comandos no son muy flexibles ni potentes, por lo que surgen los paquetes `multicol` y `multicolpar`. El primero de ellos nos dota del comando:

```
\begin{multicols}{3}[Escribiendo en columnas]
  Esto son tres columnas a las que les hemos
  puesto una cabecera, que aparecerá sobre
  ellas. En este caso no se produce el salto de
  página ni tampoco hay que hacer nada
  especial para volver al modo de edición,
  salvo cerrar el entorno.
\end{multicols}
```

#### Escribiendo en columnas

Esto son tres columnas a las que les hemos puesto una cabecera, que aparecerá sobre ellas. En este caso no se produce el salto de página ni tampoco hay que hacer nada especial para volver al modo de edición, salvo cerrar el entorno.

Por su parte, el paquete `multicolpar` es adecuado para imprimir textos en paralelo, por ejemplo, si tenemos un texto y su traducción:

```
\begin{multicolpar}{2}
  Esta es una de las columnas, que contiene
  el texto que deseamos en español (por
  ejemplo).
\par
  This is the other column, which is written
  in the other language, (english in this
  case).
\par
\end{multicolpar}
```

Esta es una de las columnas, que contiene el texto que deseamos en español (por ejemplo).

This is the other column, which is written in the other language, (english in this case).

### 5.3. Notas a pie de página

Para incluir una nota a pie de página<sup>1</sup> en nuestros documentos, usaremos el comando `LATEX \footnote{texto}` en el punto donde queramos que se inserte la referencia.

Se puede personalizar el símbolo con el que `LATEX` marca las referencias a notas al pie para que utilice, por ejemplo, símbolos en lugar de números, de la siguiente manera:

```
\renewcommand{\thefootnote}{\fnsymbol{footnote}}
```

Éste sería el resultado <sup>\*\*</sup>.

---

<sup>1</sup>Como ésta.

<sup>\*\*</sup>Cuando insertemos una nueva nota al pie.

# Capítulo 6

## Tablas

Las tablas es un elemento algo complejo que conviene ser definido de una manera bastante detallada. La notación de  $\text{\LaTeX}$  es muy potente y por ello a primera vista puede parecer engorrosa.

### 6.1. Tablas básicas

El primer entorno que aprenderemos es el `tabular` que sirve para hacer los estadillos que forman la tabla:

```
\begin{tabular}[Posicion]{formatocolumnas}
celda11 & celda12 & ... & celda1n\\
celda21 & celda22 & ... & celda2n\\
... & ... & ... & ... \\
celdam1 & celdam2 & ... & celdamn\\
\end{tabular}
```

**Posición** Indica la posición de la tabla con respecto a la línea donde se ha incluido. Puede ser `t` (top), `b` (botton), `c` (línea base).

**&** Sirve para marcar el finalizado de una celda.

**\\** Sirve para finalizar una fila.

**Formato de columnas** Podremos incluir varios ejemplares de cada uno de los símbolos que se indican a continuación. Según los que pongamos  $\text{\LaTeX}$  creará columnas de formatos diferentes:

**l** Columna alineada a la izquierda.

**r** Columna alineada a la derecha.

**c** Columna centrada.

**p{ancho}** Columna de un ancho fijo, hay que indicar la medida usada por ejemplo `p{5cm}`.

### 6.1.1. Líneas y separadores

Para introducir líneas o separadores entre columnas deberemos introducir el caracter `|` entre los símbolos indicados en el formato de columnas lo que dibujaría una línea vertical (con dos la línea sería doble) o bien `@{objeto}` donde el objeto sería un símbolo que se introduciría siempre entre las columnas (por ejemplos un punto que nos puede servir para mostrar los números decimales de una forma muy vistosa).

Para introducir líneas horizontales deberemos usar el comando `\hline` entre las diferentes filas lo que iría dibujando rayas.

### 6.1.2. Comando extracolsep

Nos va a permitir construir un separador que consiste en un espacio vacío de una longitud dada.

`@{\extracolsep{longitud}}` % todas las longitudes deben indicar unidades

### 6.1.3. Multicolumn

Este comando nos permitirá unir celdas de la misma fila.

`\multicolumn{Número}{Formato Columna}{Contenido de la nueva celda}`

**Número** Será el número de celdas a unir en la nueva celda

**Formato Columna** Son los valores incluidos en el formato de columnas de tabular (l, r, c).

Hay que tener claro que cuando usemos `multicolumn` muchas veces tendremos que incluir un `|` para que se dibuje el trozo de raya vertical correspondiente.

`\multicolumn{2}{c|}{Contenido}`

### 6.1.4. Rayas horizontales sobre determinadas celdas

Si queremos hacer líneas horizontales que abarquen sólo determinadas celdas deberemos usar el comando `cline`.

```
\begin{tabular}{ll@{\extracolsep{12pt}}ll}
\multicolumn{4}{c}{\textbf{Datos}} \\
\hline
\multicolumn{2}{c}{2000} & \multicolumn{2}{c}{2001} \\
\cline{1-2} \cline{3-4}
0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 \\
2 & 4 & 2 & 8 \\
3 & 9 & 3 & 27 \\
\end{tabular}
```

Datos			
2000		2001	
0	0	0	0
1	1	1	1
2	4	2	8
3	9	3	27

## 6.2. Los cuadros

$\text{\LaTeX}$  define este tipo de elementos flotantes para que destaquemos nuestras tablas. Las tablas que se introduzcan en un entorno `table` serán maquetadas de una forma más vistosa, podrán tener títulos y además podremos generar un índice de tablas que otra vez (tal como pasaba con el índice general) será generado automáticamente por  $\text{\LaTeX}$ .

```
\begin{table}[posición]
Contenido (debería ser un tabular)
\caption[LeyendaCorta]{Leyenda Larga}
\end{table}
```

```
\listoftables % Genera el índice de cuadros (compilar dos veces)
               % Normalmente se suele poner al final
               % Ojo no hay que ponerlo dentro de un apéndice.
```

# Capítulo 7

## Referencias, citas bibliográficas e índice de materias

En este capítulo veremos cómo trata  $\text{\LaTeX}$  cuestiones como las referencias cruzadas entre partes del documento, la bibliografía y las referencias en el texto a ítems presentes en ella, o los índices de términos.

### 7.1. Referencias

Una referencia cruzada nos sirve para ayudar al lector en su comprensión del documento que le presentamos, señalándole en un determinado momento partes del documento (su ubicación física) relevantes con respecto a la que le ocupa.

Construir una referencia cruzada en  $\text{\LaTeX}$  es tan fácil como utilizar el comando `\label{etiqueta}` en el lugar donde queremos establecer un “anclaje” o punto de referencia. El único cuidado que debemos tener es utilizar *etiquetas* únicas, con el fin de no confundir al compilador.

El punto de anclaje establecido pasará a estar asociado con la unidad estructural del documento activa en el lugar donde se ha colocado. Es decir, si está dentro de un entorno *figure* o *table*, se asociará con la figura o tabla correspondiente, si está simplemente en el contexto de una *section* o una *subsection* lo mismo, si está en el ámbito general de un *chapter*...

Una vez establecido el punto de anclaje, para señalarlo desde cualquier otra parte del documento, tenemos dos opciones:

- Hacer referencia a la unidad activa en la que se encuentra el anclaje, mediante el comando `\ref{etiqueta}`. Conseguiríamos un efecto como éste: *el capítulo 1 es una pequeña introducción donde la figura 1.1 nos muestra una captura del editor kile.*

- Hacer referencia a la ubicación física en la que se encuentra el anclaje, mediante el comando `\pageref{etiqueta}`. Conseguiríamos un efecto como éste: *el capítulo que empieza en la página 1 es una pequeña introducción donde la figura de la página 6 nos muestra una captura del editor kile.*

Con frecuencia, el uso de estas dos opciones se suele combinar: *el capítulo 1 (página 1) es una pequeña introducción donde la figura 1.1 (página 6) nos muestra una captura del editor kile.*

En caso de que nos interese recuperar el título del capítulo, la leyenda de la figura o cadena correspondiente a la unidad activa del documento relevante para la referencia, podemos usar el paquete `titleref`, que nos permite hacer uso del comando `\titleref{etiqueta}`.

El ejemplo completo resulta: *el capítulo 1, denominado “Introducción: Fundamentos de  $\LaTeX$ ” (página 1) es una pequeña introducción donde la figura “Una captura de kile” (1.1, página 6) nos muestra una captura del editor kile.*

## 7.2. Citas bibliográficas

A la hora de definir la bibliografía de un documento, existen en  $\LaTeX$  dos opciones, que estudiaremos por separado en la presente sección.

### 7.2.1. El entorno `thebibliography`

La primera opción, más básica y por tanto menos flexible y potente, pero quizás más sencilla, es la utilización del entorno  $\LaTeX$  `thebibliography`, que tiene el siguiente aspecto:

```
\begin{thebibliography}{longitud}
...
  \bibitem [Leyenda]{Etiqueta} Texto
...
\end{thebibliography}
```

donde *longitud* es una cadena de anchura mayor o igual a la máxima que va a ser utilizada en la numeración; *leyenda* es un parámetro opcional que representa la identificación que, ignorando la que  $\LaTeX$  crea por defecto (que es numérica), se colocará entre el texto, en el lugar donde

ubiquemos una cita, y a la derecha de la lista de referencias bibliográficas; *etiqueta* es la cadena que queremos usar para referirnos al ítem concreto y *texto* es la cita en sí (autor, título, editorial, etc.), con el formato y orden que queramos darle.

Para citar un elemento de la bibliografía desde cualquier parte del documento, usaremos el comando `\cite{etiqueta}`.

## Bibliografía

[Cita1] *L<sup>A</sup>T<sub>E</sub>X*, una imprenta en sus manos. Cascales Salinas, Bernardo et al. Editorial ADI, 2000.

... hay muchos y buenos libros sobre L<sup>A</sup>T<sub>E</sub>X [Cita1] para aprender y llegar a ser un experto...

... hay muchos y buenos libros sobre \LaTeX{} \cite{libroLatex} para aprender y llegar a ser un experto...

```
\begin{thebibliography}{9}
  \bibitem [Cita1]{libroLatex}
    \emph{\LaTeX{}}, una imprenta en sus manos}.
    \textsf{Cascales Salinas, Bernardo et al.}
    Editorial ADI, 2000.
\end{thebibliography}
```

Cuadro 7.1: Ejemplo de uso del entorno `thebibliography`

### 7.0.1. BIB<sub>T</sub>E<sub>X</sub>

Como hemos visto, el entorno `thebibliography` deja al autor la completa libertad (y responsabilidad) de definir el formato de la bibliografía. Esto puede ser una ventaja, pero también un engorroso inconveniente, sobre todo si pensamos en reordenar los ítems, en la consistencia entre ellos, etc.

Como alternativa, pues, a `thebibliography`, contamos con el programa `BIBTEX`, creado por Oren Patashnik, que se integra con `LATEX` para generar automáticamente un entorno `thebibliography` siguiendo una serie de reglas de formato establecidas.

Lo primero que hay que hacer para usar `BIBTEX` es definir la *base de datos* de citas bibliográficas que queremos incluir en nuestro documento. Esto es tan sencillo como editar un fichero con extensión `.bib` que tiene un formato concreto: contiene una serie de *registros* especiales, uno por referencia. Hay varios tipos de registros posibles, uno por cada tipo de referencia que se puede incluir:

```
@BOOK{LibroLatex,
  author = {{Cascales Salinas, Bernardo} and others},
  publisher = {ADI},
  title = {\LaTeX{}}, una imprenta en sus manos},
  year = {2000}
}

@ARTICLE{armistice,
  author = {{Cabrerero, David}, {Abalde, Carlos},
           {Varela, Carlos} and {Castro, Laura}},
  title = {ARMISTICE: An Experience Developing
           Management Software with Erlang},
  journal = {Principles, Logics, and Implementations of
            High-Level Programming Languages (PLI'03)},
  month = {Agosto}
  year = {2003}
}

@MANUAL{shortLatex,
  title = {The not so short introduction to \LaTeXee},
  author = {Oetiker, Tobias}
}

@MISC{faq,
  title = {FAQ de CervanTeX},
  howpublished = {\texttt{www.tug.org/tex-archive/help/
                  es-tex-faq/FAQ-CervanTeX.pdf}}
}
```

Igual que en el caso anterior, las citas se incluyen en el texto utilizando el comando `\cite{etiqueta}`, con la particularidad de que sólo aquéllos

documentos de la base de datos que sean mencionados en un `\cite` se visualizarán en la versión compilada del documento. Si queremos que se visualice algún documento no referenciado, tendremos que indicarlo con el comando `\nocite{etiqueta}`, y si queremos que aparezcan todos los elementos presentes en la base de datos, emplearemos `\nocite*`.

Tras haber definido el fichero de bibliografía, su uso se indica incluyendo, antes del `\end{document}` las líneas:

```
\bibliography{ficheroibase}
\bibliographystyle{estilo}
```

donde *ficheroibase* es el nombre del fichero de la base de datos *sin* la extensión `.bib` y *estilo* es uno de los siguientes:

**plain** Ordena las entradas alfabéticamente y las numera. El orden que establece es: autor, año, título.

**unsrt** Igual que *plain* pero las entradas se ordenan por orden de citación en el documento.

**alpha** En lugar de numerar las entradas, como *plain*, les otorga una etiqueta basada en el nombre del autor y el año de publicación. El orden que establece es: etiqueta, autor, año y título.

**abbrv** Equivalente a *plain*, hace que las entradas sean más pequeñas al abreviar los nombres de los autores, meses y nombres de las revistas.

Una vez hecho esto, es necesario compilar la bibliografía utilizando la herramienta `bibtex`. Este programa recibe como argumento un fichero `.aux` procedente de una primera compilación de nuestro documento `LATEX`, en la que se identifican todas las citas bibliográficas a las que se ha hecho referencia. Tras el procesado con `bibtex`, pues, será necesaria una nueva compilación con `latex` para que queden resueltas:

```
$ latex midocumento(.tex)
$ bibtex midocumento(.aux)
$ latex midocumento(.tex)
```

Nótese que el fichero `.aux` que se pasa a `bibtex` es el resultado de la compilación de `midocumento`, y no el fichero de la base de datos en sí (éste será encontrado por la herramienta gracias a que está incluido en el anterior por medio del comando `\bibliography`).

## 7.1. Índice de materias

En ocasiones puede ser de utilidad disponer de un índice de términos en nuestros documentos. La manera más sencilla de generar un índice terminológico o índice de materias, es incluir el paquete `makeidx` y utilizar alguna de las variaciones del comando

```
\index{entrada}
\index{entrada!subentrada}
\index{entrada!subentrada!subsubentrada}
```

en el lugar del documento a donde queremos que se refiera la entrada del índice. Con esto, incluyendo en el preámbulo del documento la orden `\makeindex` y la orden `\printindex` en el lugar del documento donde deseamos que se imprima, sólo nos restará compilar utilizando la herramienta `makeindex`:

```
$ latex      midocumento(.tex)
$ makeindex  midocumento(.idx)
$ latex      midocumento(.tex)
```

Esto nos generará un índice de materias de apariencia similar a:

```
\index{referencias}
\index{referencias!cruzadas}
...
\index{referencias!bibliográficas}
```

También se puede conseguir la típica estructura *véase*, haciendo referencia a otra entrada del índice de materias, utilizando una pequeña variación del comando: `\index{cadena1|see{cadena2}}`.

# Capítulo 8

## Imágenes

Tradicionalmente se ha considerado que el soporte a la inclusión de imágenes en  $\text{\LaTeX}$  era uno de los defectos de este sistema de composición. Actualmente este aspecto se ha mejorado mucho con la incorporación de nuevos paquetes y controladores que dotan a  $\text{\LaTeX}$  de una potencia comparable a cualquier otro sistema actual.

Podremos hacer nuestros propios gráficos en  $\text{\LaTeX}$ <sup>1</sup> y también podremos incorporar gráficos creados con otras herramientas. Las imágenes externas serán considerados por  $\text{\LaTeX}$  como “objetos cerrados” que serán maquetados según sus dimensiones.

### 8.1. El paquete `graphicx`

Este paquete es el fundamental en  $\text{\LaTeX}$  para incorporar imágenes, su principal opción será la del controlador encargado de presentar o imprimir la imagen. Los principales controladores que se usan son `dvips` (el tradicional), `pdftex` y el más moderno: `dvipdfm`.

Debe quedar claro que el controlador de `graphics` debe corresponderse con el programa que usemos para transformar `dvi`, es decir, si usamos el controlador `dvips` deberemos usar el programa `dvips` (DVIA POSTSCRIPT) y si usamos `dvipdfm` pues convertiremos el DVIA PDFdirectamente con el programa del mismo nombre.

```
% Dentro del preámbulo
\usepackage[dvips]{graphics} % o el controlador que más nos agrade
```

---

<sup>1</sup>algo que por complejidad no cubriremos en este cursos

**dvips** Permite la incorporación de imágenes en formato PostScript (ps), PostScript encapsulado (eps) y pcx (versión 0) y bmp (blanco y negro). Éstos dos últimos formatos no son adecuados para usar porque  $\text{\LaTeX}$  no podrá leer los datos acerca del tamaño de la imagen.

**pdflatex** Admite jpg, tif, png y pdf sin ningún tipo de problemas.

**dvipdfm** Admite jpg, png y pdf. Además dvipdfm puede ser ejecutado como programa independiente y trate fichero dvi que contengan gráficos incluidos con la opción dvips.

## 8.2. El comando `includegraphics`

```
\includegraphics[ListaOpciones]{archivo}  
\includegraphics*[ListaOpciones]{archivo}
```

La diferencia es como se tratará a la imagen si tiene un tamaño mayor al espacio donde se va a representar la imagen sobresaldrá y se superpondrá con el otro espacio (a lo mejor una columna vecina) (lo que hace `includegraphics` sin `*`) o bien no podrá salir del espacio asignado (la imagen será truncada) (lo que hace `includegraphics` con `*`).

### 8.2.1. Opciones de `includegraphics`

**width=3.5cm** Indicaría una anchura de 3.5 cm.

**heigh=20mm** Indicaría una altura de 20 mm.

**keepaspectratio** Conservaría proporciones originales de la imagen.

**draft** Sólo sacaría el nombre de la imagen y marcaría la zona a ocupar por la imagen en la versión definitiva.

**archivo** nombre del archivo de la imagen con ruta absoluta o relativa con respecto al fichero  $\text{\TeX}$ .

## 8.3. Figuras

$\text{\LaTeX}$  provee el entorno `figuras` que crea un nuevo tipo de objetos flotantes análogo a los cuadros. También podremos crear un índice de figuras y además tendremos un entorno `figure` con `*` que en el caso de usar dos

columnas nos permitira que nuestra imagen abarque el ancho de ambas columnas.

```
\begin{figure}[Posición]
Contenido: debería ser un \includegraphics
\caption[Leyendacorta]{Leyenda Larga}
\end{figure}
\listoffigures
```

# Capítulo 9

## Fórmulas

Una de las mayores potencialidades de  $\text{\LaTeX}$  reside en su expresividad a la hora de escribir fórmulas y expresiones matemáticas. Su filosofía es eminentemente descriptiva, lo que lo hace bastante sencillo y fácil de recordar, como veremos en las siguientes páginas.

### 9.1. Paquetes necesarios

La mayoría de los símbolos matemáticos se incluyen gracias al paquete `amsmath`. Además, con los paquetes `latexsymb` y `amssymb` se completa la lista de todos los símbolos, operadores y delimitadores posibles.

### 9.2. El modo matemático

Hay varias formas de “iniciar” el modo matemático en  $\text{\LaTeX}$ . Podemos hacerlo de modo *inline*, es decir, en medio de un párrafo, o bien hacer que se produzca un salto de línea y la fórmula aparezca centrada y aparte:

```
entonces si sumamos $ a + b $ obtendremos...
entonces si sumamos $$ a + b $$ obtendremos...
entonces si sumamos  $a + b$  obtendremos...
entonces si sumamos
                    
$$a + b$$

obtendremos...
```

Como equivalente a la primera forma, podemos usar el entorno `math`, y el entorno `displaymath` para la segunda.

Además, si queremos que la fórmula sea numerada por  $\text{\LaTeX}$ , podemos utilizar el entorno `equation`:

```

si entonces sumamos
\begin{equation}
  a + b
\end{equation}
obtendremos...

si entonces sumamos
 $a + b$  (9.1)

obtendremos...

```

En este caso, utilizando el comando `\eqref{etiqueta}` en lugar del normal `\ref{etiqueta}` para referirnos a una ecuación a la que hayamos etiquetado con un `\label{etiqueta}`,  $\text{\LaTeX}$  sustituirá la referencia por el número que identifica a la fórmula, entre paréntesis.

### 9.3. Exponentes y subíndices

Algo tan habitual como exponentes (superíndices) y subíndices son extremadamente sencillos de escribir en  $\text{\LaTeX}$ :

```

a_1 = b^2
a_2 = b^3
...
a_{n+1} = b^{(n+2)}

```

$$\begin{aligned}
 a_1 &= b^2 \\
 a_2 &= b^3 \\
 &\dots \\
 a_{n+1} &= b^{(n+2)}
 \end{aligned}$$

### 9.4. Fracciones y binomios

También el tratamiento de fracciones y binomios es simple e intuitivo en  $\text{\LaTeX}$ :

`\frac{2}{3} = \dfrac{4}{6} = \tfrac{6}{9}`

$$\frac{2}{3} = \frac{4}{6} = \frac{6}{9}$$

Como vemos, `\tfrac` se utiliza para obtener las fracciones en un tamaño más reducido ( $\frac{6}{9}$ ); la utilidad de `\dfrac` consiste en mostrar las fracciones ( $\frac{4}{6}$ ) a tamaño más grande cuando se utilizan entre el texto, ya que `\frac` respeta las proporciones ( $\frac{2}{3}$ ).

El tratamiento de los binomios es totalmente análogo:

`\binom{2}{3} \neq \dbinom{4}{6} \neq \tbinom{6}{9}`

$$\binom{2}{3} \neq \binom{4}{6} \neq \binom{6}{9}$$

## 9.5. Raíces, integrales, sumatorios, límites

Con sencillos comandos,  $\text{\LaTeX}$  permite representar raíces de cualquier tipo, así como integrales, sumatorios, productorios...

`\sqrt[3]{a + b^2}`  
`+ \int x dx`  
`+ \oint \dfrac{1}{x} dx`  
`+ \iiint \dfrac{a}{\sqrt{x+b}} dx`

$$\sqrt[3]{a + b^2} + \int x dx + \oint \frac{1}{x} dx + \iiint \frac{a}{\sqrt{x+b}} dx$$

`\sum_{i = 0}^n a_i \cdot b_i`  
`+ \sum_{\substack{j = 0 \\ j < i}}^n a_i`  
`+ \prod_{k = 0}^{\substack{k \geq i \\ k \leq j}} b_j`  
`+ \lim_{x \to \infty} \dfrac{x^2}{1-x}`

$$\sum_{i=0}^n a_i \cdot b_i + \sum_{\substack{j=0 \\ j < i}}^n a_i + \prod_{k=0}^{\substack{k \geq i \\ k \leq j}} b_j + \lim_{x \rightarrow \infty} \frac{x^2}{1-x}$$

## 9.6. Delimitadores

Antes vimos cómo los paréntesis alrededor de una fracción que va en medio del texto pueden no ser del tamaño adecuado:  $(\frac{2}{3})$ . Pero también puede ocurrirnos si queremos delimitar expresiones en fórmulas aparte:

$$[\int \frac{1}{x} dx + \sum_{i=0}^x 2^i] \cdot \sqrt[3]{a+b^2}$$

Para hacer que L<sup>A</sup>T<sub>E</sub>X adapte el tamaño de los delimitadores de forma que sea óptimo debemos utilizar las siguientes formas:

```
\left(  
\left[  
\left{  
\left|
```

y sus correspondientes `right`. Así, conseguiremos  $(\frac{2}{3})$  o:

$$\left[ \int \frac{1}{x} dx + \sum_{i=0}^x 2^i \right] \cdot \sqrt[3]{a+b^2}$$

## 9.7. Matrices y determinantes

Para crear matrices y determinantes, L<sup>A</sup>T<sub>E</sub>X pone a nuestra disposición el entorno `array`, cuyas opciones y argumentos son idénticos a los del entorno `tabular`. Combinando este entorno con los delimitadores tal y como los hemos estudiado en la sección anterior, podemos recrear cualquier tipo de matriz o determinante:

```
\left(  
  \begin{array}{crl}  
    x & & 3 & & m+n^2 & \\\br/>    x+y & & 5 & & m-n & \\\br/>    x^z & & \sqrt{7} & & m & \\\br/>    (x+y)^{z'} & & 10 & & 1+m & \\\br/>  \end{array}  
\right)
```

$$\begin{pmatrix} x & 3 & m+n^2 \\ x+y & 5 & m-n \\ x^z & \sqrt{7} & m \\ (x+y)^{z'} & 10 & 1+m \end{pmatrix}$$

### 9.7.1. Puntos suspensivos

En frecuentes ocasiones, se necesita especificar una matriz o determinante para los que muchas de las posiciones son conocidas y por tanto, se puede abreviar su representación utilizando puntos suspensivos (`\dots{}` ó `\ldots{}`), puntos suspensivos verticales (`\vdots{}`), puntos suspensivos en diagonal (`\ddots{}`) o puntos suspensivos centrados (`\cdots{}`):

```
\left|
  \begin{array}{cccc}
    a_{11} & & a_{12} & & \cdots & & a_{1n} & \\
    a_{21} & & a_{22} & & \cdots & & a_{2n} & \\
    \vdots & & & & \ddots & & \vdots & \\
    a_{n1} & & a_{n2} & & \cdots & & a_{nn} & 
  \end{array}
\right|
```

$$\begin{vmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{vmatrix}$$

## 9.8. Sistemas de ecuaciones

Ante un bloque de ecuaciones, tenemos dos cuestiones que resolver:

1. Cómo representar un sistema de ecuaciones
2. Cómo representar la resolución de una ecuación en más de un paso

La primera de ambas es sencilla, utilizando de manera combinada lo que hemos visto hasta ahora, junto con un pequeño nuevo truco, los comandos `\left.` y `\right.:`

```
| x | = \left\{ \begin{array}{rclcc}
& x & \& \& \text{si} & \& \& x \geq 0 \\
& -x & \& \& \text{si} & \& \& x < 0
\end{array} \right.
```

$$|x| = \begin{cases} x & \text{si } x \geq 0 \\ -x & \text{si } x < 0 \end{cases}$$

Aunque para este estilo de definiciones con casos también tenemos el útil entorno `cases`:

```
P_\alpha = \begin{cases}
\alpha & \text{si } \$\alpha\$ \text{ es impar} \\
-\alpha & \text{si } \$\alpha\$ \text{ es par} \\
1 & \text{si } \$\alpha\$ \text{ es } 0
\end{cases}
```

$$P_\alpha = \begin{cases} \alpha & \text{si } \alpha \text{ es impar} \\ -\alpha & \text{si } \alpha \text{ es par} \\ 1 & \text{si } \alpha \text{ es } 0 \end{cases}$$

En cuanto a la segunda, aunque podríamos hacerlo utilizando lo que ya sabemos, disponemos del entorno `eqnarray`, que nos facilita las cosas:

$$(a + b)^2 - (a - b)^2 = \tag{9.2}$$

$$(a^2 + 2ab + b^2) - (a^2 - 2ab + b^2) = 4ab \tag{9.3}$$

Además, como vemos, igual que en el entorno `equation`, cada paso es numerado automáticamente.

## 9.9. Teoremas, lemas, corolarios y demostraciones

Cuando tratamos con teoremas, lemas o corolarios, deseamos que aparezcan destacados, sin tener por ello que repetir el mismo trabajo cada vez. Incluyendo el paquete `amsthm`,  $\text{\LaTeX}$  permite utilizar el comando:

```
\newtheorem{nombreEntorno}{nombreTipo}
```

donde *nombreEntorno* es el nombre con el que nos referiremos a un nuevo entorno creado por nosotros para estos menesteres y *nombreTipo* puede ser Lema, Teorema, Corolario o Conjetura:

```

\newtheorem{teorema}{Teorema}
\newtheorem{corolario}{Corolario}
\begin{teorema}[Bolzano]
  Si  $f(x)$  es continua en un intervalo cerrado  $[a,b]$  y
  toma valores de signo opuesto en los extremos,
  entonces existe un punto  $c$  perteneciente al intervalo
  abierto  $(a,b)$  tal que  $f(c)=0$ .
  $$
  \exists \quad c \in (a, b) \quad / \quad f(c) = 0
  $$
\end{teorema}
\begin{corolario}
  El valor  $c$  es una raíz de la ecuación  $f(x) = 0$ .
\end{corolario}

```

**Teorema 1 (Bolzano).** *Si  $f(x)$  es continua en un intervalo cerrado  $[a, b]$  y toma valores de signo opuesto en los extremos, entonces existe un punto  $c$  perteneciente al intervalo abierto  $(a, b)$  tal que  $f(c) = 0$ .*

$$\exists \quad c \in (a, b) \quad / \quad f(c) = 0$$

**Corolario 1.** *El valor  $c$  es una raíz de la ecuación  $f(x) = 0$ .*

Además, el mencionado paquete nos proporciona también el entorno `proof`:

*Demostración.* Se demuestra por definición de continuidad en un intervalo cerrado, y aplicando el teorema de Weierstrass. □

## 9.10. Otros elementos útiles

### 9.10.1. Funciones y símbolos

Durante los ejemplos que hemos visto a lo largo de este capítulo, hemos comprobado el uso de algunas funciones y símbolos matemáticos típicos. Por supuesto,  $\text{\LaTeX}$  posee muchos más, de manera que lo más recomendable es consultar un libro o manual cada vez que se requiera alguno.

<code>\sin</code>	<code>\cos</code>	<code>\tan</code>	<code>\log</code>	<code>\partial</code>	
<code>\alpha</code>	<code>\beta</code>	<code>\epsilon</code>	<code>\theta</code>	<code>\lambda</code>	<code>\pi</code>
<code>\sigma</code>	<code>\phi</code>	<code>\omega</code>	<code>\beta</code>	<code>\mu</code>	<code>\gamma</code>
<code>\rho</code>	<code>\tau</code>	<code>\delta</code>	<code>\eta</code>		
<code>\Theta</code>	<code>\Lambda</code>	<code>\Omega</code>	<code>\Gamma</code>	<code>\Delta</code>	<code>\Phi</code>
<code>\approx</code>	<code>\equiv</code>	<code>\simeq</code>	<code>\cong</code>	<code>\propto</code>	
<code>\rightarrow</code>	<code>\leftarrow</code>	<code>\forall</code>	<code>\exists</code>	<code>\emptyset</code>	
<code>\lceil x</code>	<code>\rceil</code>	<code>\lfloor x</code>	<code>\rfloor</code>	<code>\subset</code>	<code>\subseteq</code>

sin	cos	tan	log	$\partial$
$\alpha$	$\beta$	$\epsilon$	$\theta$	$\lambda$ $\pi$
$\sigma$	$\phi$	$\omega$	$\beta$	$\mu$ $\gamma$
	$\rho$	$\tau$	$\delta$	$\eta$
$\Theta$	$\Lambda$	$\Omega$	$\Gamma$	$\Delta$ $\Phi$
$\approx$	$\equiv$	$\simeq$	$\cong$	$\propto$
$\Rightarrow$	$\Leftrightarrow$	$\forall$	$\exists$	$\emptyset$
$\lceil x$	$\rceil$	$\lfloor x$	$\rfloor$	$\subset$ $\subseteq$

### 9.10.2. Texto y fuentes

Como también hemos visto ya en algún ejemplo, la forma de incluir texto normal en el entorno matemático es usar el comando `\text{texto}`.

Además, existen varios tipos de fuentes que pueden usarse:

**`\mathrm`**

que produce este “Resultado =  $a + b$ ”.

**`\mathnormal`**

que produce este “*Resultado =  $a + b$* ”.

**`\mathsf`**

que produce este “Resultado =  $a + b$ ”.

**`\mathit`**

que produce este “*Resultado =  $a + b$* ”.

**`\mathbf`**

que produce este “**Resultado =  $a + b$** ”.

**`\mathcal`**

que produce este “ $\mathcal{R} = a + b$ ”.

**`\mathhtt`**

que produce este “Resultado =  $a + b$ ”.

### 9.10.3. Espacios

Para que una fórmula o ecuación quede exactamente como queremos, puede ser necesario incluir o eliminar espacios entre operadores, operandos, delimitadores y símbolos. Para ellos contamos con:

Comandos para incluir espacios en orden creciente de anchura introducida:

```
\thinspace  
\medspace  
\thickspace  
\quad  
\qquad  
\hspace{long}
```

Comandos para reducir espacio en orden creciente de anchura eliminada:

```
\negthinspace  
\negmedspace  
\negthickspace
```

### 9.10.4. Cajas

Ya para terminar, mencionaremos el comando `\boxed{fórmula}`, que nos permite presentar una fórmula encuadrada:

$$\boxed{\lim_{x \rightarrow \infty} \frac{x^2}{1-x}}$$

El mundo matemático de  $\text{\LaTeX}$  es enormemente grande. Por ello, la selección incluida aquí es una pequeña muestra de todo lo que puede hacerse. Existen muchos otros comandos y variaciones de los aquí presentados, que tan sólo buscan ser un acicate para que el lector novel interesado investigue por su cuenta, con un poco de base. Si algún lector familiarizado con  $\text{\LaTeX}$  no ha encontrado aquí su comando favorito, le pedimos disculpas por ello O:-).

# Capítulo 10

## Más sobre fuentes

$\LaTeX$  debido al diseño que Knuth hizo en  $\TeX$  arrastraba ciertos problemas con el tema de uso de fuentes. En concreto  $\TeX$  usaba unos tipos de letra determinados<sup>1</sup> y tenía problemas para incorporar otros distintos lo que hacía que los documentos tuvieran todos una estética algo repetitiva que algunos calificaban como anticuada. Hoy en día con el uso de nuevos paquetes esto no constituye ningún problema y con  $\LaTeX$  podremos usar fuentes `Postscript` y `truetype` sin ningún tipo de problema.

### 10.1. La codificación

La primera traba que imponía  $\TeX$  era el uso de una peculiar codificación en las fuentes `Computer Modern` que sólo tenían 128 símbolos (llamada codificación `OT1`), los básicos en idiomas occidentales. Lo que obligaba a que caracteres peculiares de ciertos idiomas (ñ, tildes, ...) se tuvieran que componer mediante la unión de símbolos básicos (una ñ era una n más una ligadura).

Hoy en día eso está superado gracias a los paquetes `inputenc` que se encarga de codificar adecuadamente símbolos con codificación `latin1` (tabla de códigos normal en español) a la codificación que usara el sistema  $\LaTeX$  sin que el usuario tuviera que intervenir.

El otro paquete que elimina este problema es `fontenc` que directamente hace que  $\LaTeX$  trabaje con tablas de códigos de 256 caracteres (codificación `T1`), lo que elimina ciertos problemas de composición que podía tener el compilador al unir caracteres. Así que resumiendo si no tenemos tener problemas con símbolos especiales de idiomas deberemos introducir

---

<sup>1</sup>Fuentes `Computer Modern`: definían tipos con letra redonda, serif y espaciado además de las matemáticas

en nuestro preámbulo las siguientes ordenes (consultar las opciones para otros idiomas distintos del español o gallego):

```
\usepackage[T1]{fontenc}
\usepackage[latin1]{inputenc}
```

## 10.2. ¿Por qué el pdf se ve mal en Acrobat Reader?

Este es un problema que se suele echar en cara a  $\LaTeX$  lo primero que hay que aclarar es que en realidad no es problema sino que es una consecuencia de como trabajan ambos programas.

### 10.2.1. $\LaTeX$ y su generación de tipos de letra

No vamos a profundizar en el tema de como se construyen las fuentes lo que nos llevaría al lenguaje `metafont` (otra criatura de Knuth) algo que se excede del objetivo de este curso pero hay que entender como construye  $\TeX$  un documento.

El compilador  $\TeX$  lo único que hace es ir encajando cajas en las páginas, todo en  $\LaTeX$  (una letra, una imagen, una tabla, ...) es una caja con unas dimensiones determinadas que el compilador encajará de la mejor manera posible en la página.

Por ello lo único que necesita saber  $\LaTeX$  acerca de la fuente que usamos son sus medidas (altura, anchura y profundidad). El contenido de la caja (el símbolo a imprimir) es mostrado por la herramienta que usamos para visualizar el fichero `dvi`.

Por ello las fuentes  $\LaTeX$  suelen tener al menos un par de ficheros:<sup>2</sup> uno que indica las medidas de cada símbolo y otro con los algoritmos adecuados para general el símbolo, es decir, las fuentes que usamos son vectoriales (no dependen de la resolución usada en la impresión del documento).

El visor `textttdvi` será el encargado de llamar a `metafont` y generar los símbolos. Es aquí donde se generan fuentes escalares. El visor `dvi` indicará una resolución dada (a lo mejor 600 ppp) y será la que muestre en pantalla y además la que se use para transformar el `dvi` a `ps` o a `pdf`.

---

<sup>2</sup>la explicación exacta es más compleja pero a nivel de usuario no se necesita saber más

Por eso cuando alguien mire nuestro documento desde `acrobat` estará viendo fuentes escalares (que si dependen de la resolución para imprimir) y además debido a la pésima implementación de estos tipos de letra (fuentes tipo 3) en `acrobat` estas letras se ven horrible.

Es curioso ver que si el intercambio de documento se hiciera en formato `dvi` (que al menos en principio es independiente de plataforma) no hubiera ocurrido este problema debido a que el visor `dvi` se hubiera encargado de generar la resolución correcta. Por desgracia el formato `pdf` es el habitual para el intercambio de documentos en Internet y tendremos que buscar otra solución al problema.

La solución pasa porque nuestra conversión a `pdf` se haga mediante fuentes tipo 1 (vectoriales), habrá que obligar a nuestro generador de `dvi` que use esas fuentes. Un ejemplo para `dvips` es el siguiente

```
% dvips -Pcmz -Pamz -o mydoc.dvi mydoc.tex
```

### 10.2.2. Usar fuentes PostScript

Las fuentes `PostScript` es el estándar de facto en la edición profesional, imprentas e incluso impresoras de alta calidad. Adobe es la empresa que definió este estándar que incluye 35 fuentes (Times-roman, Palatino, Bookman, Courier, ...). A mayores ésta y otras empresas han creado nuevos tipos (ficheros `tfm`).

En este curso sólo vamos a ver como usar alguna de las 35 fuentes estándar algo que es muy sencillo pues sólo tendremos que incorporar alguno de estos paquetes. La primera fila de la tabla incluye el comportamiento por defecto de `LATEX`. Se pueden incorporar diferentes paquetes que irán alterando las diferentes familias estándar.

Hay que hacer notar que estas fuentes están disponibles en Linux porque una empresa las construyó bajo `metafont` y las liberó para beneficio de la comunidad.

<b>Paquete</b>	<code>rmfamily</code>	<code>sffamily</code>	<code>ttfamily</code>	Matemáticas
<code>mathptmx</code> <code>mathpazo</code> <code>helvet</code> <code>avant</code> <code>chancery</code> <code>bookman</code> <code>newcent</code> <code>courier</code>	CM Roman Times Roman Palatino  Zapf Chancery Bookman NewCenturySchoolbook	CM sanserif  Helvetica AvantGarde  AvantGarde AvantGarde	CM Typewriter   Courier Courier Courier	CM Roman Times Roman Palatino
<code>times</code> <code>Palatino</code>	Times Roman Palatino	Helvetica Helvetica	Courier Courier	

# Capítulo 11

## L<sup>A</sup>T<sub>E</sub>X y el hipertexto

En este capítulo trataremos aspectos relacionados con L<sup>A</sup>T<sub>E</sub>X y HTML.

### 11.1. Exportar a HTML

Son muchos los programas que se pueden usar a la hora de exportar un documento L<sup>A</sup>T<sub>E</sub>X a HTML: `hyperlatex`, `tex4ht`, `latex2html`...

No obstante, `hyperlatex` implementa sólo un subconjunto de L<sup>A</sup>T<sub>E</sub>X, así que nos centraremos en `tex4ht` y `latex2html`.

#### 11.1.1. `tex4ht`

Para usar la herramienta `tex4ht`, simplemente debemos incluir el paquete del mismo nombre en el preámbulo de nuestro documento. Luego, durante el proceso de compilación, debemos ejecutar:

```
latex midocumento
tex4ht midocumento
t4ht -ddirDestino/ midocumento
```

#### 11.1.2. `latex2html`

En cuanto a `latex2html`, su uso es igualmente sencillo. En este caso el paquete a incluir se llama `html`, y no necesitamos compilar el documento L<sup>A</sup>T<sub>E</sub>X para obtener la versión HTML, puesto que `latex2html` realiza la conversión desde el código fuente, y no a partir del DVI, como es el caso de `tex4ht`:

```
latex2html -dir dirDestino -split +1 +white midocumento
```

# Capítulo 12

## Presentaciones

Tanto en el mundo educativo como en el profesional hoy en día es común que tengamos que hacer presentaciones donde exponer nuestros trabajos e ideas. Con la abundancia actual de medios no es raro que podamos tener que llegar a usar un proyector que nos permita mostrar la pantalla de nuestro ordenador a nuestra audiencia.

Son varias las alternativas para preparar transparencias, ahora bien, nosotros como usuarios de  $\text{\LaTeX}$  deberíamos intentar seguir usándolo para reutilizar el código de nuestra ponencia (el texto completo que hemos escrito).

Con la ayuda de diversos paquetes podremos llegar a realizar transparencia de una altísima calidad, así como crear diferentes efectos entre transparencias (vistosidad para el público) así como tener hiperenlaces o menús que nos ayuden durante la oratoria. Hay que señalar que la clase de documento original de  $\text{\LaTeX}$  para esta tarea (slides) hoy por hoy ha sido abandonada. Por último decir que el formato de salida de nuestras creaciones será el PDF por ser el más adecuado para estas labores.

Este año presentaremos la clase **prosper** que por su gran sencillez permite empezar a trabajar en este tipo de documentos de una manera muy rápida. Lo primero que hay que tener claro es que prosper es una clase de documento al igual que book, article o report.

### 12.1. Esquema de la clase prosper

```
\documentclass[opciones]{prosper}
\title{Título de la Presentación}
\subtitle{Subtítulo}
\author{Autor1, Autor2}
```

```

\email{email}
\institution{Empresa o Institución}
\Logo(x,y){LogoCorporativo} % x,y posición relativa
                                % a la esquina infer izqda.
\Logo{LogoCorporativo}
\slideCaption{Anotaciones y leyenda}
\displayVersion

\begin{document}
\maketitle

\begin{slide}[Transición]{Título de la pantalla}
Contenido de la primera transparencia
\end{slide}

\overlays{Número de animaciones}{
\begin{slide}

\end{slide}}
\end{document}

```

## 12.2. Estilos de la presentación

Una de las opciones de la clase prosper es el estilo, en principio tenemos 12 a escoger y hay muchos más circulando por Internet. Variando el estilo obtendremos diferentes tipos de diseños visuales.

Estilos básicos: alienglow, autumn, azure, contemporain, darkblue, frames, lignesbleues, nuancegris, troisponts, gyom, pascal y rico.

## 12.3. Más opciones de la clase prosper

**ps o pdf** Podemos desear que la salida sea a ps porque las transparencias pueden ser mostradas en retroproyector.

**draft o final** la opción draft indica mucha más información.

**total o nototal** con total se indica el número de página junto el total de páginas.

**slideBW** restringe el uso de colores usado para que quede bien impreso en blanco y negro.

**colorBG** o **nocolorBG** con **nocolorBG** el fondo de la pantalla siempre es blanco aunque el estilo escogido indique otra cosa.

**accumulate** o **noaccumulate** Prosper permite crear animaciones en base a superponer páginas. Con **noaccumulate** se permiten esas animaciones con **accumulate** todas las páginas se acumulan en una.

## 12.4. Como hacer las transparencias

A cada slide le podremos asignar un efecto de transición que será uno de los definidos en el estandar PDF(Blinds, Box, Dissolve, Glitter, Replace, Split, Wipe). Podemos usar el comando `\DefaultTransition{nombre}` para indicar que transición queremos usar salvo que indiquemos otra cosa.

El contenido de la clase prosper serán las transparencias definidas por cada uno de los entornos slide más la transparencia del título (que hace el comando `maketitle`).

El contenido de un entorno slide se representa en una sola página, si no cabe habrá una parte que no será visible.

Si queremos dividir la presentación en diferentes apartados se hará con la orden:

```
\part[transición]{Título de la Sección}
```

El contenido habitual de cada slide será entornos `itemize` por lo que éste se ha redefinido y ya no justifica por la derecha (evita separaciones de palabras).

### 12.4.1. Otros comandos

```
% Comandos para varias los dos tipos usados
\FontTitle{FuenteColor}{FuenteBlancoNegro}
\FontText{FuenteColor}{FuenteBlanconegro}
% Declara el color a utilizar en las leyendas
\ColorFoot{Color}
% Declara el tipo de letra a utilizar
\fontTitle{texto} % usar fuente del título en ese texto
\fontText{texto} % usar fuente normal para ese texto
```

# Bibliografía

- [1] *ΛT<sub>E</sub>X: una imprenta en sus manos.*  
Bernardo Cascales Salinas et al.  
ADI, 2000.
- [2] *Text Processing using ΛT<sub>E</sub>X*  
Tim Love, Richard Prager.  
<http://www-h.eng.cam.ac.uk/help/tpl/textprocessing/>
- [3] *Getting Started with ΛT<sub>E</sub>X*  
David R. Wilkins.  
<http://www.maths.tcd.ie/~dwilkins/LaTeXPrimer/Index.html>
- [4] *ΛT<sub>E</sub>X para Linux Debian con ejemplos prácticos*  
Paco Aldarias Raya  
<http://www.iespana.es/heberg/pub.htm?Nom=pacodebian>
- [5] *Documentación de TeTeX*  
tetex-doc: paquete común en todas las distribuciones Linux
- [6] *Lista de Correo GPUL-ΛT<sub>E</sub>X* <http://ceu.fi.udc.es/cgi-bin/mailman/listinfo/>
- [7] *Grupo de Programadores y Usuarios de Linux* <http://www.gpul.org>  
Próximamente: <http://latex.gpul.org>