

# LaTeX y Git

aplicado a la investigación científica

11 al 15 Enero 2016  
**FACULTAD DE CIENCIAS**

**50 HORAS**

25 horas Presenciales  
25 horas NO Presenciales

**75 euros**

**Plazas limitadas**

Impartido por Ángel Pablo Hinojosa  
Renato Ramírez Rivero

Técnicos de la Oficina de Software Libre de la Universidad de Granada



**Solicitados CRÉDITOS DE LIBRE CONFIGURACIÓN**

VER EN WEB MÁS INFORMACIÓN SOBRE LAS TITULACIONES  
SOLICITADAS Y/O RECONOCIDAS



Organizado por



Colabora



+ Info:

[www.darwineventur.es](http://www.darwineventur.es)  
[info@darwineventur.es](mailto:info@darwineventur.es)

# La introducción no-tan-corta a L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub>

---

*o L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> en 147 minutos*

por Tobias Oetiker

Hubert Partl, Irene Hyna y Elisabeth Schlegl

Versión 5.03: Agosto 2014

Traducción de carleos@uniovi.es

Actualizaciones de daniel.cuevas@cryptolab.net

jlrn77@gmail.com

Copyright ©1995-2014 Tobias Oetiker y colaboradores. Todos los derechos reservados.

Este documento es libre; puede distribuirlo o modificarlo bajo los términos de la Licencia Pública General de GNU versión 2 o (a su elección) cualquier versión posterior, publicada por la Free Software Foundation.

Este documento se distribuye con la esperanza de que sea útil, pero SIN NINGUNA GARANTÍA; incluso sin la garantía implícita de COMERCIALIZACIÓN o APTITUD PARA UN PROPÓSITO PARTICULAR. Véase la Licencia Pública General de GNU para más detalles.

Debería haber recibido una copia de la Licencia Pública General de GNU junto con este documento; si no, escriba a la Free Software Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, Estados Unidos.

# ¡Gracias!

Mucho material usado en esta introducción proviene de una introducción austriaca a L<sup>A</sup>T<sub>E</sub>X 2.09 escrita en alemán por:

Hubert Partl    <partl@mail.boku.ac.at>

*Zentraler Informatikdienst der Universität für Bodenkultur Wien*

Irene Hyna    <Irene.Hyna@bmwf.ac.at>

*Bundesministerium für Wissenschaft und Forschung Wien*

Elisabeth Schlegl    <sin~electrocorreo>

*en Graz*

Si está interesado en el documento alemán, puede encontrar una versión actualizada para L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> por Jörg Knappen en

[CTAN:/tex-archive/info/lshort/german](http://CTAN:/tex-archive/info/lshort/german)

Para la traducción al español, se han tomado muchas ideas de la versión 0.4b de Tomás Bautista. David Pérez contribuyó a corregir muchos errores tipográficos. Carlos Carleos revisó y amplió la traducción en 2008. Enrique Lazcorreta, Luis Rivera y Daniel Vela hicieron una actualización en 2014.

Los siguientes individuos ayudaron con correcciones, sugerencias y material a mejorar este texto. Hicieron un gran esfuerzo para ayudarme a poner este documento en su forma actual. Me gustaría sinceramente agradecerse a todos ellos. Naturalmente, todos los errores que encuentre en este libro son míos. Si encuentra alguna vez alguna palabra escrita correctamente, debe de ser de alguna de las personas listadas a continuación.

Rosemary Bailey, Marc Bevand, Friedemann Brauer, Jan Busa, Markus Brühwiler, Pietro Braione, David Carlisle, José Carlos Santos, Neil Carter, Mike Chapman, Pierre Chardaire, Christopher Chin, Carl Cerecke, Chris McCormack, Wim van Dam, Jan Dittberner, Michael John Downes, Matthias Dreier, David Dureisseix, Elliot, Hans Ehrbar, Daniel Flipo, David Frey, Hans Fugal, Robin Fairbairns, Jörg Fischer, Erik Frisk, Mic Milic Frederickx, Frank, Kasper B. Graversen, Arlo Griffiths, Alexandre Guimond, Andy Goth, Cyril Goutte, Greg Gamble, Frank Fischli, Morten Høgholm, Neil Hammond, Rasmus Borup Hansen, Joseph Hilferty, Björn Hvittfeldt, Martien Hulsen, Werner Icking, Jakob, Eric Jacoboni, Alan Jeffrey, Byron Jones, David Jones, Johannes-Maria Kaltenbach, Michael Koundouros, Andrzej Kawalec, Sander de Kievit, Alain Kessi, Christian Kern, Tobias Klauser, Jörg Knappen, Kjetil Kjernsmo, Maik Lehardt, Rémi Letot, Flori Lambrechts, Axel Liljencrantz, Johan Lundberg, Alexander Mai, Hendrik Maryns, Martin Maechler, Aleksandar S Milosevic, Henrik Mitsch, Claus Malten, Kevin Van Maren, Richard Nagy, Philipp Nagele, Lenimar Nunes de Andrade, Manuel Oetiker, Urs Oswald, Martin Pfister, Demerson Andre Polli, Nikos Pothitos, Maksym Polyakov Hubert Partl, John Reffing, Mike Ressler, Brian Ripley, Young U. Ryu, Bernd Rosenlecher, Chris Rowley, Risto Saarelma, Hanspeter Schmid, Craig Schlenter, Gilles Schintgen, Baron Schwartz, Christopher Sawtell, Miles Spielberg, Geoffrey Swindale, Laszlo Szathmary, Boris Tobotras, Josef Tkadlec, Scott Veirs, Didier Verna, Fabian Wernli, Carl-Gustav Werner, David Woodhouse, Chris York, Fritz Zaucker, Rick Zacccone, Mikhail Zotov y Álvaro Jaramillo Duque.

# Prefacio

L<sup>A</sup>T<sub>E</sub>X [1] es un sistema de composición muy adecuado para realizar documentos científicos y matemáticos de alta calidad tipográfica. Es también adecuado para producir documentos de cualquier otro tipo, desde simples cartas a libros enteros. L<sup>A</sup>T<sub>E</sub>X emplea T<sub>E</sub>X [2] como motor de formato.

Esta breve introducción describe L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> y debería bastar para la mayoría de las aplicaciones de L<sup>A</sup>T<sub>E</sub>X. Consulte [1, 3] para una descripción exhaustiva del sistema L<sup>A</sup>T<sub>E</sub>X.

Esta introducción se divide en 6 capítulos:

**El capítulo 1** trata sobre la estructura básica de documentos L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub>.

Aprenderá un poco sobre la historia de L<sup>A</sup>T<sub>E</sub>X. Tras leer este capítulo, debería tener un conocimiento somero de cómo trabaja L<sup>A</sup>T<sub>E</sub>X.

**El capítulo 2** profundiza en los detalles como componer los documentos.

Explica la mayoría de las órdenes y entornos esenciales de L<sup>A</sup>T<sub>E</sub>X. Tras leer este capítulo, debería ser capaz de escribir sus primeros documentos.

**El capítulo 3** explica cómo componer fórmulas con L<sup>A</sup>T<sub>E</sub>X. Con muchos ejemplos se muestra cómo usar uno de los puntos fuertes de L<sup>A</sup>T<sub>E</sub>X. Al final del capítulo hay tablas con todos los símbolos matemáticos disponibles en L<sup>A</sup>T<sub>E</sub>X.

**El capítulo 4** explica los índices, generación de bibliografías e inclusión de gráficos EPS. Presenta la creación de documentos PDF mediante pdfL<sup>A</sup>T<sub>E</sub>X y varios paquetes adicionales interesantes.

**El capítulo 5** muestra cómo usar L<sup>A</sup>T<sub>E</sub>X para crear gráficos. En lugar de dibujar una figura con algún programa gráfico, grabarla en un fichero y después incluirla en L<sup>A</sup>T<sub>E</sub>X, podrá describir directamente el dibujo L<sup>A</sup>T<sub>E</sub>X lo dibujará por usted.

**El capítulo 6** contiene información potencialmente peligrosa sobre cómo alterar la presentación normal del documento producido con L<sup>A</sup>T<sub>E</sub>X. Le indicará cómo cambiar cosas de forma que la salida hermosa de L<sup>A</sup>T<sub>E</sub>X se volverá horrible o deslumbrante, según sus habilidades.

Es importante leer los capítulos en orden —el libro no es tan largo, después de todo—. Asegúrese de leer con cuidado los ejemplos, porque mucha información está en los ejemplos dispersos a lo largo del libro.

L<sup>A</sup>T<sub>E</sub>X está disponible para la mayor parte de ordenadores, desde PC y Mac a grandes sistemas UNIX y VMS. En muchos ordenadores universitarios encontrará una instalación de L<sup>A</sup>T<sub>E</sub>X disponible y lista para usar. Habrá información sobre cómo acceder la instalación local de L<sup>A</sup>T<sub>E</sub>X en la *Local Guide* [5]. Si tiene problemas para comenzar, pregunte a la persona que le proporcionó este libro. El objetivo de este documento *no* es contarle cómo instalar y configurar un sistema L<sup>A</sup>T<sub>E</sub>X, sino enseñarle cómo escribir documentos para que pueda procesarlos con L<sup>A</sup>T<sub>E</sub>X.

Si necesita conseguir cualquier material relativo a L<sup>A</sup>T<sub>E</sub>X, eche un vistazo a las páginas de la Red Integral de Ficheros T<sub>E</sub>X (CTAN). La página de internet se encuentra en <http://www.ctan.org>. Todos los paquetes pueden conseguirse desde la dirección ftp <ftp://www.ctan.org> y sus espejos en todo el mundo.

Encontrará otras referencias a CTAN a lo largo del libro, especialmente indicaciones a programas y documentos que podría querer descargar. En lugar de escribir direcciones completas, sólo escribí CTAN: seguido del lugar dentro de CTAN al que debería acceder.

Si quiere ejecutar L<sup>A</sup>T<sub>E</sub>X es su propio ordenador, busque qué hay disponible en <CTAN:/tex-archive/systems>.

Si se le ocurre qué puede añadirse, eliminarse o cambiarse en este documento, por favor hágamelo saber. Estoy especialmente interesado en opiniones de novatos en L<sup>A</sup>T<sub>E</sub>X sobre qué partes de esta intro son fáciles de entender y cuáles deberían explicarse mejor.

Tobias Oetiker    [<oetiker@ee.ethz.ch>](mailto:oetiker@ee.ethz.ch)

Departamento de Tecnología de la Información e  
Ingeniería Eléctrica,  
Instituto Federal Suizo de Tecnología

La versión actual de este documento está disponible en  
<CTAN:/tex-archive/info/lshort>

# Índice general

<b>¡Gracias!</b>	<b>III</b>
<b>Prefacio</b>	<b>v</b>
<b>1. Cosas que debe saber</b>	<b>1</b>
1.1. El nombre del hombre . . . . .	1
1.1.1. $\text{\TeX}$ . . . . .	1
1.1.2. $\text{\LaTeX}$ . . . . .	1
1.2. Lo básico . . . . .	2
1.2.1. Autor, maquetador y compositor . . . . .	2
1.2.2. Maquetación . . . . .	2
1.2.3. Ventajas y desventajas . . . . .	3
1.3. Ficheros de entrada $\text{\LaTeX}$ . . . . .	4
1.3.1. Espacio . . . . .	4
1.3.2. Caracteres especiales . . . . .	5
1.3.3. Órdenes $\text{\LaTeX}$ . . . . .	5
1.3.4. Comentarios . . . . .	6
1.4. Estructura del fichero de entrada . . . . .	7
1.5. Una típica sesión de consola o línea de órdenes . . . . .	7
1.6. El aspecto del documento . . . . .	9
1.6.1. Clases de documento . . . . .	9
1.6.2. Paquetes . . . . .	10
1.6.3. Estilos de página . . . . .	12
1.7. Ficheros que puede encontrarse . . . . .	12
1.8. Proyectos grandes . . . . .	14
<b>2. Composición de texto</b>	<b>17</b>
2.1. La estructura del texto y el idioma . . . . .	17
2.2. Saltos de línea y de página . . . . .	19
2.2.1. Justificación de párrafos . . . . .	19
2.2.2. Silabación . . . . .	20
2.3. Cadenas a medida . . . . .	21
2.4. Símbolos y caracteres especiales . . . . .	21



2.4.1.	Comillas	21
2.4.2.	Guiones y rayas	22
2.4.3.	Tilde	22
2.4.4.	Slash (/)	22
2.4.5.	Símbolo de grado ( $\circ$ )	23
2.4.6.	El símbolo monetario del euro ( $\text{€}$ )	23
2.4.7.	Puntos suspensivos (...)	23
2.4.8.	Ligaduras	24
2.4.9.	Acentos y caracteres especiales	24
2.5.	Soporte para otros idiomas	25
2.5.1.	Soporte para el castellano	28
2.5.2.	La opción Unicode	33
2.6.	El espacio entre palabras	37
2.7.	Títulos, capítulos y secciones	37
2.8.	Referencias cruzadas	40
2.9.	Notas al pie	40
2.10.	Palabras enfatizadas	41
2.11.	Entornos	41
2.11.1.	Listas ( <code>itemize</code> , <code>enumerate</code> y <code>description</code> )	42
2.11.2.	Alineación ( <code>flushleft</code> , <code>flushright</code> y <code>center</code> )	42
2.11.3.	Citas ( <code>quote</code> , <code>quotation</code> y <code>verse</code> )	43
2.11.4.	Resumen ( <code>abstract</code> )	43
2.11.5.	Citas literales ( <code>verbatim</code> )	44
2.11.6.	Tablas ( <code>tabular</code> )	44
2.12.	Elementos deslizantes	46
2.13.	Protección de órdenes frágiles	49
<b>3.</b>	<b>Composición de fórmulas matemáticas</b>	<b>51</b>
3.1.	Generalidades	51
3.2.	Agrupación en modo matemático	53
3.3.	Construcción de bloques de una fórmula matemática	53
3.4.	Espaciado en matemáticas	58
3.5.	Material alineado verticalmente	59
3.6.	Fantasmas	60
3.7.	Tamaño de fundición en matemáticas	61
3.8.	Lemas, teoremas, corolarios, ...	62
3.9.	Símbolos en negrita	64
3.10.	Lista de símbolos matemáticos	65
<b>4.</b>	<b>Especialidades</b>	<b>73</b>
4.1.	Inclusión de Encapsulated POSTSCRIPT	73
4.2.	Bibliografía	75
4.3.	Índices	77
4.4.	Cabeceras personalizadas	78

4.5.	El paquete Verbatim . . . . .	80
4.6.	Instalación de paquetes adicionales . . . . .	80
4.7.	Uso de pdfL <sup>A</sup> T <sub>E</sub> X . . . . .	81
4.7.1.	Documentos PDF para la red . . . . .	82
4.7.2.	Las fundiciones . . . . .	83
4.7.3.	Uso de gráficos . . . . .	85
4.7.4.	Enlaces de hipertexto . . . . .	85
4.7.5.	Problemas con enlaces . . . . .	88
4.7.6.	Problemas con marcadores . . . . .	88
4.8.	Trabajo con X <sub>Y</sub> L <sup>A</sup> T <sub>E</sub> X . . . . .	90
4.8.1.	Las fundiciones . . . . .	90
4.8.2.	Compatibilidad entre X <sub>Y</sub> L <sup>A</sup> T <sub>E</sub> X y pdfL <sup>A</sup> T <sub>E</sub> X . . . . .	92
4.9.	Creación de presentaciones . . . . .	92
<b>5.</b>	<b>Producción de gráficos matemáticos</b>	<b>97</b>
5.1.	Panorama general . . . . .	97
5.2.	El entorno picture . . . . .	98
5.2.1.	Órdenes básicas . . . . .	98
5.2.2.	Segmentos de recta . . . . .	100
5.2.3.	Flechas . . . . .	101
5.2.4.	Circunferencias y círculos . . . . .	102
5.2.5.	Texto y fórmulas . . . . .	103
5.2.6.	\multiput y \linethickness . . . . .	103
5.2.7.	Óvalos . . . . .	104
5.2.8.	Uso múltiple de cajas de dibujos predefinidas . . . . .	105
5.2.9.	Curvas de Bézier cuadráticas . . . . .	106
5.2.10.	Catenaria . . . . .	107
5.2.11.	Rapidez en la Teoría Especial de la Relatividad . . . . .	108
5.3.	Los paquetes graficos PGF y TikZ . . . . .	108
<b>6.</b>	<b>Personalización de L<sup>A</sup>T<sub>E</sub>X</b>	<b>113</b>
6.1.	Nuevas órdenes, entornos y paquetes . . . . .	113
6.1.1.	Órdenes nuevas . . . . .	114
6.1.2.	Nuevos entornos . . . . .	115
6.1.3.	Espacio extra . . . . .	115
6.1.4.	Línea de órdenes L <sup>A</sup> T <sub>E</sub> X . . . . .	116
6.1.5.	Su propio paquete . . . . .	116
6.2.	Fundiciones y tamaños . . . . .	117
6.2.1.	Órdenes que cambian la fundición . . . . .	117
6.2.2.	¡Atención: peligro! . . . . .	120
6.2.3.	Consejo . . . . .	121
6.3.	Espaciado . . . . .	121
6.3.1.	Espacio entre renglones . . . . .	121
6.3.2.	Formato de párrafo . . . . .	121

6.3.3. Espacio horizontal . . . . .	122
6.3.4. Espacio vertical . . . . .	123
6.4. Composición de la página . . . . .	124
6.5. Más diversión con las longitudes . . . . .	126
6.6. Cajas . . . . .	127
6.7. Líneas y puntales . . . . .	129
<b>Bibliografía</b>	<b>131</b>
<b>Index</b>	<b>134</b>

# Índice de figuras

1.1. Un fichero $\text{\LaTeX}$ mínimo. . . . .	7
1.2. Ejemplo de un artículo de revista. . . . .	8
4.1. Ejemplo de configuración de <code>fancyhdr</code> . . . . .	79
4.2. Código de ejemplo para la clase <code>beamer</code> . . . . .	94
6.1. Paquete de ejemplo. . . . .	117
6.2. Parámetros de composición de la página. . . . .	125



# Índice de cuadros

1.1. Clases de documento. . . . .	10
1.2. Opciones de clases de documento. . . . .	11
1.3. Algunos paquetes distribuidos con L <sup>A</sup> T <sub>E</sub> X. . . . .	13
1.4. Los estilos de página predefinidos de L <sup>A</sup> T <sub>E</sub> X. . . . .	13
2.1. Una recopilación de euros . . . . .	24
2.2. Acentos y caracteres especiales. . . . .	25
2.3. Preámbulo para documentos en castellano. . . . .	28
2.4. Abreviaciones . . . . .	29
2.5. Opciones globales del castellano . . . . .	32
2.6. Permisos de deslizamiento. . . . .	47
3.1. Acentos en modo matemático. . . . .	65
3.2. Letras griegas. . . . .	65
3.3. Relaciones binarias. . . . .	66
3.4. Operadores binarios. . . . .	66
3.5. Operadores GRANDES. . . . .	67
3.6. Flechas. . . . .	67
3.7. Delimitadores. . . . .	67
3.8. Delimitadores grandes. . . . .	68
3.9. Símbolos variados. . . . .	68
3.10. Símbolos no matemáticos. . . . .	68
3.11. Delimitadores AMS. . . . .	68
3.12. Símbolos AMS griegos y hebreos. . . . .	68
3.13. Relaciones binarias AMS. . . . .	69
3.14. Flechas AMS. . . . .	70
3.15. Relaciones binarias y flechas negadas AMS. . . . .	71
3.16. Operadores binarios AMS. . . . .	71
3.17. Variados AMS. . . . .	72
3.18. Alfabetos para matemático. . . . .	72
4.1. Nombres de claves para el paquete <code>graphicx</code> . . . . .	74
4.2. Ejemplos de sintaxis de las claves para el índice. . . . .	77

6.1. Fundiciones. . . . .	118
6.2. Tamaños de fundición. . . . .	118
6.3. Tamaños absolutos en puntos para las clases normales. . . . .	119
6.4. Fundiciones para mates. . . . .	119
6.5. Unidades T <sub>E</sub> X. . . . .	123

# Capítulo 1

## Cosas que debe saber

La primera parte de este capítulo presenta un vistazo breve de la filosofía e historia de  $\text{\LaTeX}$  2<sub>ε</sub>. La segunda parte se centra en la estructura básica de un documento  $\text{\LaTeX}$ . Tras leer este capítulo, debería tener un conocimiento básico de cómo funciona  $\text{\LaTeX}$ , que necesitará para entender el resto de este libro.

### 1.1. El nombre del hombre

#### 1.1.1. $\text{\TeX}$

$\text{\TeX}$  es un programa de ordenador creado por Donald E. Knuth [2]. Sirve para componer texto y fórmulas matemáticas. Knuth empezó a escribir el motor de composición  $\text{\TeX}$  en 1977 para investigar el potencial de los equipos de impresión digital que estaban empezando a usarse en la industria tipográfica en aquel tiempo; en concreto tenía la esperanza de poder revertir la tendencia de calidad tipográfica en declive que él vio afectar a sus propios libros y artículos. El programa  $\text{\TeX}$  tal como lo conocemos hoy día fue publicado en 1982, con algunas sutiles mejoras añadidas en 1989 para soportar caracteres de 8 bites y múltiples lenguajes.  $\text{\TeX}$  tiene fama de ser muy estable, muy portable y prácticamente sin errores. El número de versión de  $\text{\TeX}$  converge hacia  $\pi$  y es ahora 31415926.

$\text{\TeX}$  se pronuncia “Tej”. La “j” surge del alfabeto griego donde X es la letra “j” o “ji”.  $\text{\TeX}$  es también la primera sílaba de la palabra griega  $\tau\epsilon\xi\nu\eta$  (arte). En un entorno ASCII,  $\text{\TeX}$  se convierte en  $\text{\TeX}$ .

#### 1.1.2. $\text{\LaTeX}$

$\text{\LaTeX}$  es un paquete de macros que permite a los autores componer e imprimir su trabajo con la mayor calidad tipográfica posible, usando un formato profesional predefinido.  $\text{\LaTeX}$  fue escrito originalmente por Leslie Lamport [1]. Emplea el formateador  $\text{\TeX}$  como motor de composición. Actualmente un equipo de programadores da mantenimiento a  $\text{\LaTeX}$ .



L<sup>A</sup>T<sub>E</sub>X se pronuncia “Látej”. Si quiere referirse a L<sup>A</sup>T<sub>E</sub>X en un entorno ASCII, escriba LaTeX. L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> se pronuncia “Látej dos e” y se escribe LaTeX2e.

## 1.2. Lo básico

### 1.2.1. Autor, maquetador y compositor

Para publicar algo, los autores dan su manuscrito mecanografiado a una editorial. Uno de sus maquetadores decide el aspecto del documento (anchura de columna, tipografías, espacio ante y tras cabeceras, ...). El maquetador escribe sus instrucciones en el manuscrito y luego se lo da al compositor o cajista, quien compone el libro siguiendo esas instrucciones.

Un maquetador humano suele interpretar qué pretendía el autor mientras escribía el manuscrito. Decide sobre las cabeceras de los capítulos, las citas, los ejemplos, las fórmulas, etc. basándose en su conocimiento profesional y en el contenido del manuscrito.

En un entorno L<sup>A</sup>T<sub>E</sub>X, L<sup>A</sup>T<sub>E</sub>X representa el papel del maquetador y usa T<sub>E</sub>X como su compositor. Pero L<sup>A</sup>T<sub>E</sub>X es “sólo” un programa, y por tanto necesita más supervisión. El autor tiene que proporcionar información adicional para describir la estructura lógica de su trabajo. Tal información se escribe entre el texto como “órdenes L<sup>A</sup>T<sub>E</sub>X”.

Esto es bastante diferente del enfoque visual o WYSIWYG<sup>1</sup> que sigue la mayoría de los procesadores de texto modernos, como *Abiword* u *Open/Libre-Office Writer*. Con estos programas, los autores especifican el aspecto del documento interactivamente mientras escriben texto en el ordenador. Así pueden ver en la pantalla cómo aparecerá el trabajo final cuando se imprima.

Cuando use L<sup>A</sup>T<sub>E</sub>X no suele ser posible ver el aspecto final del texto mientras lo escribe, pero tal aspecto puede verse en pantalla tras procesar el fichero mediante L<sup>A</sup>T<sub>E</sub>X. Entonces pueden hacerse correcciones antes de enviar el documento a la impresora para tener una copia en papel.

### 1.2.2. Maquetación

La maquetación (diseño tipográfico) es un arte. Los autores sin habilidad a menudo cometen errores de formato al suponer que maquetar es mayormente una cuestión de estética —“Si un documento luce bien artísticamente, está bien diseñado”—. Pero como un documento se escribe para ser leído y no colgado en una galería de arte, su legibilidad es mucho más importante que su aspecto. Ejemplos:

- El tamaño de los tipos y la numeración de las cabeceras debe escogerse para que la estructura de capítulos y secciones quede clara al lector.

---

<sup>1</sup>What you see is what you get: lo que ve es lo que consigue.

- La longitud de línea debe ser suficientemente corta para no cansar a los ojos del lector, pero suficientemente larga para llenar la página apropiadamente.

Con sistemas WYSIWYG, los autores a menudo generan documentos agradables estéticamente pero con muy poca o muy inconsistente estructura.  $\text{\LaTeX}$  impide tales errores de formato forzando al autor a declarar la estructura *lógica* del documento.  $\text{\LaTeX}$  escoge entonces la composición más adecuada.

### 1.2.3. Ventajas y desventajas

Cuando gente del mundo WYSIWYG se encuentra con usuarios de  $\text{\LaTeX}$ , a menudo discuten “las ventajas de  $\text{\LaTeX}$  sobre un procesador de textos normal” o lo contrario. Lo mejor que puede hacer cuando un debate tal comienza es mantenerse al margen, pues tales discusiones a menudo se salen de quicio. Pero a veces uno no puede escapar...

Pues he aquí algo de munición. Las principales ventajas de  $\text{\LaTeX}$  sobre procesadores de texto normales son las siguientes:

- Se dispone de composiciones diseñadas profesionalmente, lo que hace que un documento parezca realmente “impreso”.
- El soporte para la composición de fórmulas matemáticas es muy adecuado.
- Los usuarios sólo tienen que aprender unas pocas órdenes fáciles de entender, que especifican la estructura lógica del documento. Casi nunca necesitan preocuparse del aspecto real del documento.
- Es fácil generar incluso estructuras complejas, como notas al pie, referencias, índices o bibliografías.
- Existen paquetes libres (incluso gratuitos) que facilitan muchas tareas tipográficas especializadas, no soportadas directamente por el  $\text{\LaTeX}$  básico. Por ejemplo, hay disponibles paquetes para incluir gráficos o para componer bibliografías según normas precisas. Se describen muchos de estos paquetes en *The  $\text{\LaTeX}$  Companion* [3].
- $\text{\LaTeX}$  incita a los autores a escribir textos bien estructurados, porque así trabaja  $\text{\LaTeX}$  —especificando la estructura—.
- $\text{\TeX}$ , el motor de formateo de  $\text{\LaTeX}$  2 $\epsilon$ , es libre y muy portable. Por tanto, puede ejecutarse en casi cualquier plataforma informática disponible.

$\text{\LaTeX}$  tiene también algunas desventajas, y supongo que me es un poco difícil encontrar alguna notable, aunque estoy seguro de que otros le podrán hablar de cientos ;-)

- $\text{\LaTeX}$  no funciona bien para quienes han vendido su alma a ciertas compañías...
- Aunque pueden ajustarse algunos parámetros dentro de una cierta composición del documento, el diseño de una nueva composición completa es difícil y lleva mucho tiempo.<sup>2</sup>
- Es muy duro escribir documentos desestructurados y desorganizados.
- Puede que su aprendiz nunca llegue a entender, a pesar de ciertos primeros pasos prometedores, a comprender el concepto de Mercado Lógico.

### 1.3. Ficheros de entrada $\text{\LaTeX}$

La entrada para  $\text{\LaTeX}$  es un fichero de texto puro. Puede crearlo con cualquier editor de texto. Contiene el texto del documento, así como las órdenes que dirán a  $\text{\LaTeX}$  cómo componer el texto.

#### 1.3.1. Espacio

$\text{\LaTeX}$  trata los caracteres “en blanco”, tales como el espacio en blanco o el tabulador, uniformemente como “espacio”. *Varios caracteres consecutivos en blanco se tratan como un solo “espacio”*. Espacio en blanco al principio de una línea se ignora en general, y un salto de línea aislado se trata como “espacio en blanco”.

Una línea vacía entre dos líneas de texto define el fin de un párrafo. *Varias líneas vacías se tratan igual que una sola línea vacía*. El texto de abajo es un ejemplo. A la izquierda está el texto del fichero de entrada, y a la derecha está la salida formateada.

No importa si usted deja  
uno o varios      espacios  
tras una palabra.

Una línea vacía comienza  
un nuevo párrafo.

No importa si usted deja uno o varios es-  
pacios tras una palabra.

Una línea vacía comienza un nuevo párrafo.

<sup>2</sup>Un rumor dice que esto es uno de los elementos clave que serán tratados en el futuro sistema  $\text{\LaTeX}3$ .

## 1.3.2. Caracteres especiales

Los siguientes símbolos son caracteres reservados que o tienen un significado especial bajo  $\text{\LaTeX}$  o no están disponibles en todas las tipografías. Si los pone directamente en su texto, normalmente no se imprimirán, sino que obligarán a  $\text{\LaTeX}$  a hacer cosas que usted no pretendía.

# \$ % ^ & \_ { } ~ \

Como verá, se pueden usar estos caracteres en sus documentos añadiendo una antebarra (barra invertida) como prefijo:

`\# \$ \% \^{} \& \_ \{ \} \~{} \`

# \$ % ^ & \_ { } ~

Los demás símbolos y muchos más pueden imprimirse con órdenes especiales en fórmulas matemáticas o como acentos. El carácter antebarra `\` *no* puede introducirse añadiendo otra antebarra delante (`\\`); esta secuencia se usa para saltar de línea.<sup>3</sup>

1.3.3. Órdenes  $\text{\LaTeX}$ 

Las órdenes  $\text{\LaTeX}$  son sensibles a mayúsculas, y adoptan uno de los dos formatos siguientes:

- Comienzan con una antebarra `\` y luego tienen un nombre que consiste sólo en letras. Los nombres de orden terminan con un espacio, un número o cualquier otra ‘no-letra’.
- Consisten en una antebarra y exactamente una no-letra.

$\text{\LaTeX}$  prescinde del espacio en blanco tras las órdenes. Si quiere conseguir un espacio tras una orden, tiene que poner o `{ }` y un blanco o una orden especial de espaciado tras el nombre de la orden. Las llaves `{ }` impiden a  $\text{\LaTeX}$  “comerse” todo el espacio tras el nombre de la orden.

He leído que Knuth divide a la gente que trabaja con `\TeX{}` en `\TeX{}`nicos y `\TeX` pertos.`\\`  
Hoy es `\today`.

He leído que Knuth divide a la gente que trabaja con `\TeX` en `\TeX`nicos y `\TeX`pertos.  
Hoy es 26 de agosto de 2014.

Algunas órdenes requieren un parámetro, que tiene que ponerse entre llaves `{ }` tras el nombre de la orden. Algunas órdenes soportan parámetros opcionales, que se añaden tras el nombre de la orden entre corchetes `[ ]`.

<sup>3</sup>Pruebe la orden `\backslash$` en su lugar. Produce una ‘\’.

Los siguientes ejemplos usan algunas órdenes  $\text{\LaTeX}$ . No se preocupe por ellos; se explicarán más adelante.

¡Puede `\textsl{fiarse}` de mí!

¡Puede *fiarse* de mí!

Por favor, comienza una nueva línea ¡justo aquí!`\newline`  
¡Gracias!

Por favor, comienza una nueva línea ¡justo aquí!  
¡Gracias!

#### 1.3.4. Comentarios

Cuando  $\text{\LaTeX}$  encuentra un carácter `%` al procesar un fichero de entrada, prescinde del resto de la línea actual, el salto de línea y todo el espacio en blanco al comienzo de la línea siguiente.

Esto puede usarse para escribir notas en el fichero de entrada, que no se mostrarán en la versión impresa.

Este es un `%` estúpido  
`%` Mejor: instructivo <----  
ejemplo: Supercal%  
                    ifragilíst%  
                    icoespialidoso

Este es un ejemplo: Supercalifragilísticoes-  
pialidoso

El carácter `%` también puede usarse para dividir líneas largas en la entrada donde no se permiten espacios ni saltos de línea.

Para comentarios más largos puede usar `el entorno comment` proporcionado por los paquetes `comment` o `verbatim`. Esto significa que tiene que añadir la línea `\usepackage{verbatim}` o `\usepackage{comment}` al preámbulo de su documento, como se explica abajo, antes de que pueda usar esta orden.

Este es otro  
`\begin{comment}`  
bastante estúpido,  
pero útil  
`\end{comment}`  
ejemplo para empotrar  
comentarios en su texto.

Este es otro ejemplo para empotrar comen-  
tarios en su texto.

Tenga en cuenta que eso no funciona dentro de entornos complejos, como por ejemplo los matemáticos.

## 1.4. Estructura del fichero de entrada

Cuando  $\text{\LaTeX} 2_{\epsilon}$  procesa un fichero de entrada, espera que siga una cierta estructura. Así, todo fichero de entrada ha de comenzar con la orden

```
\documentclass{...}
```

Esto indica qué tipo de documento pretende usted escribir. Después, puede incluir órdenes que influyen el estilo de todo el documento, o puede cargar paquetes que añaden nuevas prestaciones al sistema  $\text{\LaTeX}$ . Para cargar un paquete use la orden

```
\usepackage{...}
```

Cuando todo el trabajo de preparación está hecho, comience a escribir el cuerpo del texto con la orden

```
\begin{document}
```

El área entre `\documentclass` y `\begin{document}` se llama *preámbulo*.

Ahora escriba el texto mezclado con órdenes  $\text{\LaTeX}$  útiles. Al final del documento añada la orden

```
\end{document}
```

que dice a  $\text{\LaTeX}$  que termine el trabajo. Cualquier cosa que siga a esta orden será ignorada por  $\text{\LaTeX}$ .

La Figura 1.1 muestra el contenido de un fichero  $\text{\LaTeX} 2_{\epsilon}$  mínimo. Un fichero de entrada algo más complejo aparece en la Figura 1.2.

## 1.5. Una típica sesión de consola o línea de órdenes

Como se insinuaba antes (ver 1.2.1, p. 2)  $\text{\LaTeX}$  por sí mismo viene sin GUI (interfaz gráfica de usuario) ni botones para pulsar. Es un programa

```
\documentclass{article}
\usepackage[spanish]{babel}
\usepackage[latin1]{inputenc}
\begin{document}
Gracián: Lo bueno, si breve...
```

Figura 1.1: Un fichero  $\text{\LaTeX}$  mínimo.

de procesamiento por lotes que “mastica”, “traga” y “digiere” su fichero de entrada para “excretar” su(s) fichero(s) de salida. Algunas instalaciones de  $\text{\LaTeX}$  ofrecen una interfaz gráfica donde usted puede escribir y compilar su fichero de entrada ( $\text{\TeX}$ nicCenter,  $\text{\TeX}$ maker, Kile). En otros sistemas puede requerirse la escritura de ciertas órdenes, de modo que he aquí cómo lograr que  $\text{\LaTeX}$  compile su fichero de entrada en un sistema basado en texto. Téngalo en cuenta: esta descripción supone que su ordenador ya dispone de una instalación de  $\text{\LaTeX}$  funcional.

1. Edite/Cree su fichero de entrada  $\text{\LaTeX}$ . Este fichero debe ser texto puro. Puede crearlo con cualquier editor de texto: vi, emacs, Nano, Gedit, Kate, etc. También puede usar un procesador de texto (Open/Libre-Office Writer, Kword, Abiword), pero asegúrese de que guarda el fichero con formato *Texto plano*. Al escoger un nombre para el fichero, póngale como extensión `.tex`.
2. Ejecute  $\text{\LaTeX}$  en su fichero de entrada. Si tiene éxito aparecerá un fichero `.dvi`. Puede que necesite ejecutar  $\text{\LaTeX}$  varias veces para que los índices y todas las referencias internas queden correctamente definidas. Si su fichero de entrada tiene un error  $\text{\LaTeX}$  se lo dirá y parará el procesamiento de su fichero de entrada. Escriba `ctrl-D` para volver a la línea de órdenes.

```
latex mi-documento.tex
```

```
\documentclass[a4paper,11pt]{article}
% define el título
\author{H.-Partl}
\title{Minimalismo}
\begin{document}
% genera el título
\maketitle
% inserta el índice general
\tableofcontents
\section{Algunas palabras interesantes}
Y bien, aquí comienza mi articulillo.
\section{Adiós, Mundo}
...y aquí termina.
\end{document}
```

Figura 1.2: Ejemplo de un artículo de revista. Todas las órdenes que ve en este ejemplo se explicarán más tarde.

3. Ahora puede visualizar el fichero DVI. Hay varias maneras de hacerlo. Puede mostrar el fichero en pantalla con

```
xdvi mi-documento.dvi &
```

Esto funciona en GNU o Unix con X11. En ReactOS o Windows puede probar **yap** (yet another previewer).

También puede convertir el fichero dvi a POSTSCRIPT para imprimirlo o visualizarlo con Ghostscript.

```
dvips -Pcmz mi-documento.dvi -o mi-documento.ps
```

Su sistema  $\text{\LaTeX}$  puede incluir las herramientas `dvipdf` o `dvipdfm`, que le permiten convertir el fichero `.dvi` directamente en pdf.

```
dvipdf mi-documento.dvi
```

Finalmente,  $\text{\PDFLaTeX}$  le permite compilar el fichero directamente en pdf.

```
pdflatex mi-documento
```

## 1.6. El aspecto del documento

### 1.6.1. Clases de documento

La primera información que  $\text{\LaTeX}$  necesita saber cuando procesa un fichero de entrada es el tipo de documento que el autor quiere crear. Esto se indica con la orden `\documentclass`.

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

Aquí *clase* indica el tipo de documento por crear. El Cuadro 1.1 lista las clases de documentos explicadas en esta introducción. La distribución de  $\text{\LaTeX} 2_{\epsilon}$  proporciona clases adicionales para otros documentos, incluyendo cartas y diapositivas (presentaciones). El parámetro *opciones* personaliza el comportamiento de la clase. Las opciones tienen que separarse por comas. Las opciones más comunes para las clases de documento habituales se listan en el Cuadro 1.2.

Ejemplo: Un fichero de entrada para un documento  $\text{\LaTeX}$  podría empezar con la línea

```
\documentclass[11pt,twoside,a4paper]{article}
```



que manda a  $\text{\LaTeX}$  componer el documento como un *artículo* con un tamaño de fundición básica de *once puntos*, y producir un documento adecuado para imprimir a *doble cara* en *papel A4*.

### 1.6.2. Paquetes

Mientras escribe su documento, probablemente halle que hay algunas áreas donde el  $\text{\LaTeX}$  básico no puede resolver su problema. Si quiere incluir gráficos, texto en color o código fuente de un fichero en su documento, necesita mejorar las capacidades de  $\text{\LaTeX}$ . Tales mejoras se introducen con *paquetes*. Los paquetes se activan con la orden

```
\usepackage[opciones]{paquete}
```

donde *paquete* es el nombre del paquete y *opciones* es una lista de palabras clave que activan funciones especiales del paquete. Algunos paquetes vienen con la distribución básica de  $\text{\LaTeX 2}_{\epsilon}$  (vea Cuadro 1.3). Otros se proporcionan por separado. Puede encontrar más información sobre los paquetes instalados en su ordenador en la *Local Guide* [5]. La principal fuente de información sobre paquetes de  $\text{\LaTeX}$  es *The  $\text{\LaTeX}$  Companion* [3]. Contiene descripciones de cientos de paquetes, junto con información sobre cómo escribir sus propias extensiones de  $\text{\LaTeX 2}_{\epsilon}$ .

Las distribuciones modernas de  $\text{\TeX}$  vienen con un gran número de paquetes preinstalados. Si está trabajando en un sistema GNU o Unix, use

Cuadro 1.1: Clases de documento.

<b>article</b>	para artículos en revistas científicas, informes breves, documentación de programas, invitaciones, ...
<b>proc</b>	para actas, basado en la clase <i>article</i> .
<b>minimal</b>	es lo más pequeña posible. Solamente establece un tamaño de página y una fundición (tipo de letra). Se usa principalmente para depurar errores.
<b>report</b>	para informes más largos que contienen varios capítulos, pequeños libros, tesis doctorales, ...
<b>book</b>	para libros reales
<b>slides</b>	para diapositivas. La clase usa letras grandes sin serifas. También puede en su lugar usar las clases $\text{\FoilTeX}$ , <i>Prosper</i> o <i>Beamer</i> .

Cuadro 1.2: Opciones de clases de documento.

---

<code>10pt</code> , <code>11pt</code> , <code>12pt</code>	Establece el tamaño de la principal fundición del documento. Si no se especifica ninguna opción, se aplica <code>10pt</code> .
<code>a4paper</code> , <code>letterpaper</code> , ...	Define el tamaño del papel. El tamaño por omisión es <code>letterpaper</code> . Además de esas dos, pueden indicarse <code>a5paper</code> , <code>b5paper</code> , <code>executivepaper</code> , y <code>legalpaper</code> .
<code>fleqn</code>	Dispone las fórmulas destacadas hacia la izquierda en vez de centradas.
<code>leqno</code>	Coloca los números de las fórmulas a la izquierda en vez de a la derecha.
<code>titlepage</code> , <code>notitlepage</code>	Indica si tras el título del documento debe empezarse una página nueva o no. La clase <code>article</code> no comienza página nueva por omisión, mientras que <code>report</code> y <code>book</code> sí la tienen.
<code>onecolumn</code> , <code>twocolumn</code>	Dice a L <sup>A</sup> T <sub>E</sub> X que componga el documento en una columna o dos columnas respectivamente.
<code>twoside</code> , <code>oneside</code>	Indica si quiere generar el documento a dos caras o a una, respectivamente. Las clases <code>article</code> y <code>report</code> son a una cara y la clase <code>book</code> es a dos caras por omisión. Tenga en cuenta que esta opción concierne solamente al aspecto del documento. La opción <code>twoside no</code> dice a su impresora que debería de hecho imprimir a dos caras.
<code>landscape</code>	Cambia la composición del documento para imprimirlo en modo apaisado.
<code>openright</code> , <code>openany</code>	Hace que los capítulos comiencen o sólo en páginas de la derecha, o en la siguiente página disponible. Esto no funciona con la clase <code>article</code> , pues no entiende de capítulos. La clase <code>report</code> por omisión comienza capítulos en la página siguiente disponible y la clase <code>book</code> los comienza en páginas de la derecha.

---

la orden `texdoc` para acceder a información sobre paquetes.

### 1.6.3. Estilos de página

L<sup>A</sup>T<sub>E</sub>X soporta tres combinaciones predefinidas de cabeceras y pies de página, llamadas estilos de página. El parámetro *estilo* de la orden

`\pagestyle{estilo}`

define cuál emplearse. El cuadro 1.4 lista los estilos de página predefinidos. Es posible cambiar el estilo de la página actual con la orden

`\thispagestyle{estilo}`

Se puede encontrar una descripción de cómo crear sus propias cabeceras y pies en *The L<sup>A</sup>T<sub>E</sub>X Companion* [3] y en la sección 4.4 en la página 78.

## 1.7. Ficheros que puede encontrarse

Cuando trabaje con L<sup>A</sup>T<sub>E</sub>X se encontrará pronto con un batiburrillo de ficheros con extensiones variadas. La lista siguiente explica los diversos tipos de fichero que puede encontrar cuando trabaje con T<sub>E</sub>X. Tenga en cuenta que esta tabla no pretende ser una lista completa de extensiones, pero si encuentra una que piense que es importante, por favor escríbame indicándolo.

- .tex** Fichero de entrada L<sup>A</sup>T<sub>E</sub>X (o T<sub>E</sub>X). Puede compilarse con `latex` (o `tex`).
- .sty** L<sup>A</sup>T<sub>E</sub>X Paquete de macros. Es un fichero que puede cargar en su documento L<sup>A</sup>T<sub>E</sub>X usando la orden `\usepackage`.
- .dtx** T<sub>E</sub>X documentado. Es el formato principal para distribuir ficheros de estilo L<sup>A</sup>T<sub>E</sub>X. Si procesa un fichero `.dtx` obtiene código macro documentado del paquete L<sup>A</sup>T<sub>E</sub>X contenido en el fichero `.dtx`.
- .ins** El instalador para los ficheros contenidos en el fichero `.dtx` correspondiente. Si descarga un paquete L<sup>A</sup>T<sub>E</sub>X de la red, normalmente obtendrá un fichero `.dtx` y uno `.ins`. Ejecute L<sup>A</sup>T<sub>E</sub>X sobre el fichero `.ins` para desempacar el fichero `.dtx`.
- .cls** Los ficheros de clase definen el aspecto de su documento. Se seleccionan mediante la orden `\documentclass`.
- .fd** Fichero de descripción de una fundición que define nuevas fundiciones para L<sup>A</sup>T<sub>E</sub>X.

Cuadro 1.3: Algunos paquetes distribuidos con L<sup>A</sup>T<sub>E</sub>X.

---

<b>doc</b>	Permite la documentación de programas L <sup>A</sup> T <sub>E</sub> X. Descrito en <code>doc.dtx</code> <sup>a</sup> y en <i>The L<sup>A</sup>T<sub>E</sub>X Companion</i> [3].
<b>exscale</b>	Proporciona versiones escaladas de la fundición de la extensión matemática. Descrito en <code>ltexscale.dtx</code> .
<b>fontenc</b>	Indica qué codificación de fundición debería usar L <sup>A</sup> T <sub>E</sub> X. Descrito en <code>ltoutenc.dtx</code> .
<b>ifthen</b>	Proporciona órdenes de la forma ‘si...entonces...si no...’. Descrito en <code>ifthen.dtx</code> y <i>The L<sup>A</sup>T<sub>E</sub>X Companion</i> [3].
<b>latexsym</b>	Para acceder a la fundición de símbolos de L <sup>A</sup> T <sub>E</sub> X, debería usar el paquete <code>latexsym</code> . Descrito en <code>latexsym.dtx</code> y en <i>The L<sup>A</sup>T<sub>E</sub>X Companion</i> [3].
<b>makeidx</b>	Proporciona órdenes para producir índices. Descrito en la sección 4.3 y en <i>The L<sup>A</sup>T<sub>E</sub>X Companion</i> [3].
<b>syntonly</b>	Procesa un documento sin componerlo. Útil para localizar errores.
<b>inputenc</b>	Permite indicar una codificación para la entrada como ASCII, ISO Latin-1, ISO Latin-2, páginas de código 437/850 IBM, Apple Macintosh, Next, UTF-8 o una definida por el usuario. Descrito en <code>inputenc.dtx</code> .

---

<sup>a</sup>Este fichero debería estar instalado en su sistema, y usted debería ser capaz de crear el correspondiente `dvi` escribiendo `latex doc.dtx` en cualquier directorio en que tenga permiso de escritura. Lo mismo aplica para todos los demás ficheros mencionados en este cuadro.

Cuadro 1.4: Los estilos de página predefinidos de L<sup>A</sup>T<sub>E</sub>X.

---

<b>plain</b>	imprime los números de página en la parte de abajo, en el centro del pie. Es el estilo por omisión.
<b>headings</b>	imprime el nombre del capítulo actual y el número de página en la cabecera de cada página, mientras que el pie queda vacío. (Es el estilo usado en este documento)
<b>empty</b>	deja vacíos tanto la cabecera como el pie de página.

---

Los siguientes ficheros se generan cuando ejecuta  $\text{\LaTeX}$  sobre su fichero de entrada:

- .dvi** Device Independent File (fichero independiente de dispositivo). Es el principal resultado de una compilación de  $\text{\LaTeX}$ . Puede visualizar su contenido con un programa visor DVI o puede imprimirlo mediante **dvips** o una aplicación similar.
- .log** Recoge un registro detallado de qué pasó durante la última compilación.
- .toc** Almacena todas las cabeceras de sección. Es leído en la siguiente compilación para producir el índice general.
- .lof** Es como .toc pero para la lista de figuras.
- .lot** Lo mismo, para la lista de cuadros.
- .aux** Otro fichero que conserva información de una compilación a la siguiente. Entre otras cosas, el fichero .aux se usa para las referencias cruzadas.
- .idx** Si su documento contiene un índice alfabético,  $\text{\LaTeX}$  almacena todas las palabras del índice en este fichero. Procese este fichero con **makeindex**. Acuda a la sección 4.3 en la página 77 para más información sobre indexado.
- .ind** El fichero .idx procesado, listo para ser incluido en su documento en el próximo ciclo de compilaciones.
- .ilg** Registro con lo que hizo **makeindex**.

## 1.8. Proyectos grandes

Cuando trabaje en proyectos grandes, puede servirle dividir el fichero de entrada en varias partes que puede reunir al compilarlo.  $\text{\LaTeX}$  tiene dos órdenes que lo ayudan a hacerlo.

`\include{nombre-de-fichero}`

Puede usar esta orden en el cuerpo del documento para insertar el contenido de otro fichero llamado *nombre-de-fichero.tex*. Tenga en cuenta que  $\text{\LaTeX}$  comenzará una nueva página antes de procesar el material proveniente de *nombre-de-fichero.tex*.

La segunda orden puede usarse en el preámbulo. Le permite indicar a  $\text{\LaTeX}$  que solamente incluya algunos de los ficheros señalados mediante  $\text{\include}$ .

$\text{\includeonly}\{nombre-fichero-1,nombre-fichero-2,...\}$

Tras ejecutar esta orden en el preámbulo del documento, sólo se ejecutarán las órdenes  $\text{\include}$  para los ficheros listados en el argumento de la orden  $\text{\includeonly}$ . Fíjese en que no ha de haber ningún espacio entre los nombres de ficheros y las comas.

La orden  $\text{\include}$  comienza componiendo el texto incluido en una nueva página. Esto ayuda cuando usa  $\text{\includeonly}$ , porque los saltos de página no se moverán, incluso cuando se omitan algunos ficheros. A veces esto no es deseable. En tal caso, puede usar la orden

$\text{\input}\{nombre-de-fichero\}$

que simplemente incluye el fichero especificado, sin efectos especiales y sin insertar espacio adicional.

Para que  $\text{\LaTeX}$  inspeccione rápidamente su documento puede usar el paquete `syntonly`. Hace que  $\text{\LaTeX}$  recorra su documento sólo comprobando la corrección de la sintaxis y el uso de órdenes, pero no produce ninguna salida (DVI). Puesto que  $\text{\LaTeX}$  se ejecuta más rápido de este modo puede hacerle ahorrar mucho tiempo valioso. El uso es muy sencillo:

```
 $\text{\usepackage}\{syntonly\}$   
 $\text{\syntonly}$ 
```

Cuando quiera producir páginas, basta con comentar la segunda línea (mediante la adición de un signo de porcentaje al principio).



## Capítulo 2

# Composición de texto

Tras leer el capítulo previo, debería conocer lo básico para entender de qué está hecho un documento  $\text{\LaTeX} 2_{\epsilon}$ . En este capítulo se explica el resto de la estructura que se necesita saber para producir un documento útil.

### 2.1. La estructura del texto y el idioma

Por Hanspeter Schmid <[hanspi@schmid-werren.ch](mailto:hanspi@schmid-werren.ch)>

El quid de escribir un texto (salvo cierta literatura moderna) es comunicar ideas, información o conocimiento al lector. El lector entenderá mejor el texto si dichas ideas están bien estructuradas, y verá y sentirá dicha estructura mucho mejor si la forma tipográfica refleja la estructura lógica y semántica del contenido.

$\text{\LaTeX}$  se diferencia de otros sistemas de composición en que sólo tiene que decirle tal estructura. La forma tipográfica del texto se deriva según las “reglas” dadas en el fichero de clase del documento y en los varios ficheros de estilo usados.

La unidad de texto más importante en  $\text{\LaTeX}$  (y en tipografía) es el párrafo. Lo llamamos “unidad de texto” porque un párrafo es la forma tipográfica que debería reflejar un pensamiento o una idea básica completos. Así, si comienza un nuevo pensamiento, debería empezar un nuevo párrafo; y si no, deberían usarse sólo saltos de línea. Si duda sobre insertar saltos de párrafo, recuerde que su texto es un vehículo de ideas y pensamientos. Si tiene un salto de párrafo, pero el anterior pensamiento continúa, debería eliminar el salto. Si aparece una línea de pensamiento totalmente nueva en el mismo párrafo, entonces debería insertar un salto.

Casi todo el mundo subestima completamente la importancia de saltos de párrafo bien situados. Mucha gente no sabe siquiera cuál es el significado de un salto de párrafo o, especialmente en  $\text{\LaTeX}$ , introduce saltos de párrafo sin saberlo. Este último error es especialmente fácil de cometer si se usan ecuaciones en el texto. Mire los siguientes ejemplos, y piense por qué a veces



se usan líneas vacías (saltos de párrafo) antes y después de la ecuación, y a veces no. (Si no entiende bien todavía todas las órdenes para entender estos ejemplos, lea este capítulo y el siguiente y luego lea esta sección otra vez.)

```
% Ejemplo 1
...cuando Einstein presentó su fórmula
\begin{equation}
  e = m \cdot c^2 \; ,
\end{equation}
que es al mismo tiempo la fórmula física
más famosa y la menos entendida.

% Ejemplo 2
...de lo cual se sigue la ley de corrientes de Kirchhoff:
\begin{equation}
  \sum_{k=1}^n I_k = 0 \; .
\end{equation}

La ley de tensiones de Kirchhoff puede derivarse...

% Ejemplo 3
...lo que tiene varias ventajas.

\begin{equation}
  I_D = I_F - I_R
\end{equation}
es el núcleo de un modelo de transistor muy eficiente. ...
```

La siguiente unidad de texto más pequeña es la oración. En textos ingleses, hay un espacio mayor tras un punto que termina una oración que tras uno que termina una abreviatura.  $\text{\LaTeX}$  supone por omisión que un punto termina una oración; si se equivoca, debe indicarle qué es lo que desea. Esto se explicará más tarde en este capítulo. Afortunadamente, en español no afecta tanto esta distinción.

La estructuración de un texto se extiende incluso a partes de las oraciones. La mayoría de los idiomas tienen reglas de puntuación muy complicadas, pero en muchos idiomas (incluido el español) acertará casi siempre con las comas si recuerda lo que representan: una pausa breve en el flujo del lenguaje. Si no está seguro de dónde poner una coma, lea la oración en alto y tómese un breve respiro en cada coma. Si le suena mal en algún lugar, borre esa coma; si siente que le urge respirar (o hacer una breve pausa) en otro lugar, inserte una coma.

Finalmente, los párrafos de un texto deberían estar estructurados también a un nivel más alto, distribuyéndose en capítulos, secciones, subsecciones, y así sucesivamente. Sin embargo, el efecto tipográfico de escribir p.ej. `\section{La estructura del texto y el idioma}` es tan obvio que es casi evidente cómo deben usarse estas estructuras de alto nivel.

## 2.2. Saltos de línea y de página

### 2.2.1. Justificación de párrafos

Los libros se suelen componer con líneas de igual longitud.  $\text{\LaTeX}$  inserta los saltos de línea y los espacios necesarios entre palabras optimizando el contenido de todo un párrafo. Si es preciso, también divide palabras con guiones si no caben bien en una línea. Cómo se componen los párrafos depende de la clase del documento. Normalmente la primera línea de un párrafo lleva sangría, y no hay espacio adicional entre dos párrafos. Tiene más información al respecto en la sección 6.3.2.

En casos concretos puede ser necesario ordenar a  $\text{\LaTeX}$  que salte de línea:

```
\ ó \newline
```

comienza una nueva línea sin comenzar un nuevo párrafo.

```
\
```

además prohíbe un salto de página tras el salto forzado de línea.

```
\newpage
```

comienza una nueva página.

```
\linebreak[n], \nolinebreak[n], \pagebreak[n], \nopagebreak[n]
```

producen un salto de línea, impiden un salto de línea, producen un salto de página, o impiden un salto de página, respectivamene. Permiten al autor ajustar sus efectos mediante el argumento opcional  $n$ , al que puede asignarse un número entre cero y cuatro. Poniendo  $n$  a un valor menor que 4, deja a  $\text{\LaTeX}$  la opción de no hacer caso de su orden si el resultado tiene mal aspecto. No confunda estas órdenes “-break” con las órdenes “new-”. Incluso si pone una orden “-break”,  $\text{\LaTeX}$  aún intenta dejar bien el borde derecho de la página y la longitud total de la página, como se describe en la sección siguiente. Si realmente quiere iniciar una nueva línea, use la orden “newline”.

L<sup>A</sup>T<sub>E</sub>X siempre intenta producir los mejores saltos de página posibles. Si no puede encontrar una manera de dividir las líneas que cumpla con sus expectativas, permite que una línea se salga por la derecha del párrafo. L<sup>A</sup>T<sub>E</sub>X se queja entonces (“overfull hbox”) mientras procesa el fichero de entrada. Esto sucede muy a menudo cuando L<sup>A</sup>T<sub>E</sub>X no puede encontrar un lugar adecuado para dividir una palabra.<sup>1</sup> Puede mandar a L<sup>A</sup>T<sub>E</sub>X que baje sus expectativas un poco mediante la orden `\sloppy`. Impide las líneas extralargas incrementando el espaciado permitido entre palabras —aunque la salida final no sea óptima—. En tal caso se advierte al usuario (“underfull hbox”). En la mayoría de los casos el resultado no tiene muy buen aspecto. La orden `\fussy`, por el contrario, indica a L<sup>A</sup>T<sub>E</sub>X que debe ser más exigente en sus elecciones.

### 2.2.2. Silabación

L<sup>A</sup>T<sub>E</sub>X divide las palabras según sus sílabas al final del renglón si lo considera necesario. Si el algoritmo de división no encuentra los puntos de silabación correctos, puede remediar la situación usando las siguientes órdenes para decirle a T<sub>E</sub>X las excepciones. La orden

`\hyphenation{lista de palabras}`

causa que las palabras listadas en el argumento se dividan sólo en los puntos marcados con “-”. El argumento de la orden debería contener sólo palabras de letras normales o, mejor dicho, signos que L<sup>A</sup>T<sub>E</sub>X considera letras normales. Las sugerencias de silabación se almacenan para el idioma activo mientras se da la orden. Esto quiere decir que si da la orden en el preámbulo del documento entonces influirá la silabación del inglés. Si sitúa la orden tras `\begin{document}` y está usando algún paquete para otro idioma como **babel**, entonces las sugerencias de silabación estarán activas para el idioma activo de **babel**.

El ejemplo de abajo permitirá que “guiones” se divida, y también “Guiones”; e impedirá que “FORTRAN”, “Fortran” y “fortran” se dividan en ningún caso. Sólo se permiten caracteres ASCII (no las vocales acentuadas ni la ñe) en el argumento.

Ejemplo:

```
\hyphenation{FORTRAN Gui-o-nes}
```

La orden `\-` inserta un guión discrecional en una palabra, que se convierte en el único punto donde se permite la división en dicha palabra. Esta orden

---

<sup>1</sup>Aunque L<sup>A</sup>T<sub>E</sub>X le avisa cuando ocurre (Overfull hbox) y muestra la línea problemática, tales líneas no siempre son fáciles de encontrar. Si usa la opción `draft` en la orden `\documentclass`, tales líneas se marcarán con una línea negra gruesa en el margen derecho.

es útil sobre todo para palabras que contienen caracteres especiales (p.ej. vocales acentuadas), porque  $\text{\LaTeX}$  no divide automáticamente tales palabras.

Me parece que es: su\per-ca\-%  
li\fra-gi\lís\ti-co-es\-%  
pia\li-do\so

Me parece que es: supercalifragilísticoes-  
pialidoso

Para mantener varias palabras juntas en el mismo renglón use la orden

`\mbox{texto}`

que causa que su argumento quede junto en todas las circunstancias.

Mi número telefónico pasará  
a ser `\mbox{677 843 860}` a  
partir de mañana.

Mi número telefónico pasará a ser  
677 843 860 a partir de mañana.

El parámetro indicado como  
`\mbox{\emph{nombre}\filenomo{}}`  
contiene el nombre del `\filenomo{}`.

El parámetro indicado como  
*nombrefichero* contiene el nombre  
del fichero.

`\fbox` es similar a `\mbox`, pero además dibujará un rectángulo visible alrededor del argumento.

## 2.3. Cadenas a medida

En algunos ejemplos de las páginas anteriores, ha visto algunas órdenes simples de  $\text{\LaTeX}$  para componer cadenas de texto especiales:

Orden	Ejemplo	Descripción
<code>\today</code>	26 de agosto de 2014	Fecha de hoy
<code>\TeX</code>	$\text{\TeX}$	Su compositor favorito
<code>\LaTeX</code>	$\text{\LaTeX}$	El nombre del hombre
<code>\LaTeXe</code>	$\text{\LaTeX 2\epsilon}$	La encarnación actual

## 2.4. Símbolos y caracteres especiales

### 2.4.1. Comillas

No use " para las comillas como haría con una máquina de escribir. En tipografía hay comillas especiales de apertura y cierre. En  $\text{\LaTeX}$ , use dos ` (acentos graves) para abrir comillas y dos ' (apóstrofes) para cerrar comillas *inglesas*. Para comillas *inglesas* simples basta con poner una de cada una.

“Por favor, pulse la tecla ‘x’.”

“Por favor, pulse la tecla ‘x’”

Tenga en cuenta que el apóstrofo aparece en el código fuente anterior como un acento agudo (simétrico al grave).

En la tipografía española, las comillas tradicionales son « y ». La versión española debería ser así:

`\guillemotleft`

Por favor, pulse la tecla ‘‘x’’.%

`\guillemotright`

«Por favor, pulse la tecla “x”.»

### 2.4.2. Guiones y rayas

$\text{\LaTeX}$  conoce cuatro tipos de guión o raya, uno de los cuales es el signo matemático “menos”. Observe cómo obtenerlos:

`austro-húngaro, P-valor\\`

`páginas 13--67\\`

`sí ---dijo él--- \\`

`$0$, $1$ y $-1$`

austro-húngaro, P-valor

páginas 13–67

sí —dijo él—

0, 1 y −1

Los nombres de estos símbolos son: ‘-’ guión, ‘—’ raya corta, ‘—’ raya y ‘—’ signo menos. (En tipografía tradicional española, la rayacorta no existe; en su lugar se utiliza siempre el guión.)

### 2.4.3. Tilde (~)

Se trata de un carácter que aparece a menudo en código informático y direcciones de red. Para generarlo en  $\text{\LaTeX}$  puede usar `\~` pero el resultado: `\~` no es realmente lo que busca. Intente esto otro:

`http://www.rich.edu/~{}rockefeller`  
`http://www.clever.edu/$\sim$tesla`

`\http://www.rich.edu/~rockefeller`  
`http://www.clever.edu/~tesla`

### 2.4.4. Slash (/)

Para introducir una barra entre dos palabras, se puede simplemente escribir, por ejemplo, `read/write`, pero esto hace que  $\text{\LaTeX}$  trate a las dos palabras como una sola, e inhibe la división silábica en estas dos palabras, de modo que puede haber errores de cajas horizontales rebasadas (*‘overfull’*). Para evitar esto, use `\slash`. Escriba, por ejemplo, `‘read\slash write’`, lo que permite la partición silábica, pero puede aún usar el carácter de la barra normal `‘/’` para cocientes o unidades, por ejemplo: 5 MB/s.

### 2.4.5. Símbolo de grado (°)

El siguiente ejemplo muestra cómo imprimir un símbolo de grado en L<sup>A</sup>T<sub>E</sub>X:

Estamos a  
 $-30^{\circ}$ . Pronto superconduciremos.

Estamos a  $-30^{\circ}$ C. Pronto superconduciremos.

El paquete `textcomp` dispone de la orden `\textcelsius` para producir el mismo símbolo sin tener que usar superíndices (<sup>^</sup>).

### 2.4.6. El símbolo monetario del euro (€)

Si escribe sobre dinero, casi seguro que necesite el símbolo del euro. Muchas fundiciones actuales contienen el símbolo del euro. Tras cargar el paquete `textcomp` en el preámbulo de su documento

```
\usepackage{textcomp}
```

puede usar la orden

```
\texteuro
```

para acceder a él.

Si su fundición no proporciona su propio símbolo del euro o si no le gusta el símbolo de la fundición, tiene más opciones:

Primero, el paquete `eurosym`. Proporciona el símbolo oficial del euro:

```
\usepackage[official]{eurosym}
```

Si prefiere un símbolo del euro que se ajuste a su fundición, use la opción `gen` en lugar de la opción `official`.

El paquete `marvosym` también proporciona muchos y variados símbolos, incluido el del euro, con el nombre `\EUR` (y otras versiones como `\EURtm`).

### 2.4.7. Puntos suspensivos (...)

En una máquina de escribir, una coma o un punto ocupa el mismo espacio que cualquier otra letra. En tipografía, estos caracteres ocupan muy poco espacio y casi se pegan a la letra anterior. En tipografía española esto no es un problema, porque los ‘puntos suspensivos’ van casi juntos. En tipografía inglesa no, así que en lugar de escribir tres puntos use la orden

```
\ldots
```

Cuadro 2.1: Una recopilación de euros

LM+textcomp	<code>\texteuro</code>	€	€	€
eurosym	<code>\euro</code>	€	€	€
[gen]eurosym	<code>\euro</code>	€	€	€
marvosym	<code>\EUR</code>	€	€	€

Not like this ... but like this:\\  
New York, Tokyo, Budapest, \ldots

Not like this ... but like this:  
New York, Tokyo, Budapest, ...

En español sería:

Puntos en medio\... y al final:\\  
Nueva York, Tokio, Budapest...

Puntos en medio... y al final:  
Nueva York, Tokio, Budapest...

#### 2.4.8. Ligaduras

Algunas combinaciones de letras se componen no sólo poniendo una letra tras otra, sino usando símbolos especiales.

`ff fi fl ffi...` en lugar de `ff fi fl ffi ...`

Las llamadas ligaduras pueden evitarse insertando `\mbox{}` entre las dos letras en cuestión. Esto puede ser necesario para palabras compuestas de dos palabras (raro en castellano, pero común en otros idiomas).

`\Large No “\^ceffarbisto”\`  
`sino “\^cef\mbox{f}arbisto”.`

No “`^ceffarbisto`”  
sino “`^ceffarbisto`”.

#### 2.4.9. Acentos y caracteres especiales

L<sup>A</sup>T<sub>E</sub>X soporta el uso de acentos y caracteres especiales para muchos idiomas. El cuadro 2.2 muestra todo tipo de acentos aplicados a la letra o. Por supuesto también funcionan con otras letras (vocales o consonantes).

Para situar un acento sobre una i o una j, hay que quitar sus puntos. Esto se consigue escribiendo `\i` y `\j`.

```
H\^otel, na\"i ve, \'el\'eve,\\
sm\o rrebr\o d, !\'Se\~norita!,\\
Sch\"onbrunner, Schlo\ss{ },
Stra\ss e,\\
\^Ce\^ha \^sa\u umman\^ga\^j o
```

```
Hôtel, naïve, élève,
smørrebrød, ¡Señorita!,
Schönbrunner, Schloß, Straße,
Çêña šaŭmmanĝaĵo
```

Cuadro 2.2: Acentos y caracteres especiales.

ò	\‘o	ó	\’o	ô	\^o	õ	\~o
ō	\=o	ó	\.o	ö	\"o	ç	\c c
ö	\u o	ö	\v o	ő	\H o	q	\c o
q	\d o	q	\b o	ô	\t oo		
œ	\oe	Œ	\OE	æ	\ae	Æ	\AE
å	\aa	Å	\AA				
ø	\o	Ø	\O	l	\l	L	\L
i	\i	j	\j	i	!‘	¿	?‘

## 2.5. Soporte para otros idiomas

Cuando escriba documentos en idiomas distintos del español, hay tres áreas en que  $\text{\LaTeX}$  tiene que configurarse adecuadamente:

1. Todas las cadenas de texto generadas automáticamente<sup>2</sup> tienen que adaptarse al nuevo idioma. Para muchos idiomas, estos cambios pueden llevarse a cabo mediante el paquete `babel` de Johannes Braams.
2.  $\text{\LaTeX}$  necesita saber las reglas de silabación para el nuevo idioma. Crear reglas de silabación para  $\text{\LaTeX}$  es algo más difícil. Significa reconstruir el fichero de formato con patrones de silabación diferentes. Su *Local Guide* [5] debería darle más información sobre esto.
3. Reglas tipográficas específicas del idioma. En francés, por ejemplo, hay un espacio obligatorio antes de cada carácter de dos puntos (:).

<sup>2</sup>Índice general, Apéndice, ...



Si su sistema ya está configurado adecuadamente, puede activar el paquete `babel` añadiendo la orden

```
\usepackage[idioma]{babel}
```

tras la orden `\documentclass`. Puede listar los *idiomas* construidos en su sistema `LATEX` cada vez que se ejecuta el compilador. Babel activará automáticamente las reglas de silabación para el idioma que escoja. Si su formato `LATEX` no soporta la silabación del idioma escogido, babel funcionará todavía pero desactivará la silabación, lo que tiene un efecto bastante negativo en la apariencia del documento compuesto.

Babel también define nuevas órdenes para algunos idiomas, que simplifican la escritura de caracteres especiales. El idioma alemán, por ejemplo, contiene muchas diéresis (äöü). Con `babel`, puede escribir ö tecleando "o en lugar de \".

Si carga babel con múltiples idiomas

```
\usepackage[idiomaA,idiomaB]{babel}
```

entonces el último idioma en la lista de opciones será el activo (es decir, idiomaB); puede usar la orden

```
\selectlanguage{idiomaA}
```

para cambiar el idioma activo.

La mayoría de los sistemas de ordenador modernos le permiten escribir letras de diferentes alfabetos directamente desde el teclado. Para manejar varias codificaciones de entrada usadas por diferentes grupos de idiomas en diferentes plataformas `LATEX` emplea el paquete `inputenc`:

```
\usepackage[codificación]{inputenc}
```

Cuando use este paquete, debería considerar que otras personas podrían no poder editar sus ficheros de entrada en sus ordenadores, porque usan una codificación diferente. Por ejemplo, la a con diéresis ä en OS/2 tiene el código 132, en sistemas GNU o Unix que usen ISO-LATIN 1 tiene el código 228, mientras que en la codificación cirílica cp1251 para ReactOS o Windows esta letra no existe; así que use este paquete con cuidado. Las siguientes codificaciones pueden resultarle útiles, dependiendo del sistema en que esté trabajando<sup>3</sup>:

---

<sup>3</sup>Para saber más sobre codificaciones de entrada soportadas para idiomas con alfabetos latino o cirílico, lea la documentación de `inputenc.dtx` y `cyinpenc.dtx` respectivamente. La sección 4.6 explica cómo generar la documentación de los paquetes.

Sistema operativo	encodings	
	western Latin	Cyrillic
Mac	applemac	macukr
GNU, Unix	latin1	koi8-ru
Windows	ansinew	cp1251
DOS, OS/2	cp850	cp866nav

Si tiene un documento multilingüe con codificaciones que entran en conflicto, considere el uso de UNICODE a través de la codificación `utf-8`.

```
\usepackage[utf8]{inputenc}
```

le permitirá crear ficheros de entrada  $\text{\LaTeX}$  en `utf-8`, una codificación multi-octeto en que cada carácter puede ocupar desde un octeto hasta cuatro.

La codificación de fundiciones es una cuestión diferente. Define en qué posición dentro de una fundición  $\text{\TeX}$  se almacena cada letra. Múltiples codificaciones de entrada podrían corresponderse con la misma codificación de fundición, lo que reduce el número de fundiciones requeridas. Las codificaciones de fundición se manejan mediante el paquete `fontenc`:

```
\usepackage[codificación]{fontenc}
```

donde *codificación* es la codificación de fundición. Es posible cargar varias codificaciones simultáneamente.

La codificación de fundición por omisión en  $\text{\LaTeX}$  es `OT1`, la codificación de la fundición original de  $\text{\TeX}$ , Computer Modern. Contiene sólo los 128 caracteres del conjunto ASCII de 7 bits. Cuando se requieren caracteres acentuados,  $\text{\TeX}$  los crea combinando un carácter normal con un acento. Aunque el resultado parece perfecto, este enfoque impide que la silabación automática funcione en palabras que contienen caracteres acentuados. Además, algunas letras latinas no pueden crearse combinando un carácter normal con un acento; sin mencionar los casos de alfabetos no latinos, como el griego o el cirílico.

Para evitar estos inconvenientes, se crearon varias fundiciones de 8 bits similares a CM. Las fundiciones *Extended Cork* (EC) en la codificación `T1` contienen letras y signos de puntuación para la mayoría de los idiomas europeos basados en el alfabeto latino. Las fundiciones `LH` contienen letras necesarias para componer documentos en idiomas que usan el alfabeto cirílico. Dado el gran número de caracteres cirílicos, se organizan en cuatro codificaciones de fundición —`T2A`, `T2B`, `T2C` y `X2`.<sup>4</sup> El grupo `CB` contiene fundiciones en la codificación `LGR` para la composición de texto griego.

Usando estas fundiciones puede mejorar/posibilitar la silabación en documentos de otros idiomas. Otra ventaja de usar las nuevas fundiciones

<sup>4</sup>La lista de idiomas soportados por cada codificación puede hallarse en [11].

Cuadro 2.3: Preámbulo para documentos en castellano.

---

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

---

similares a CM es que proporcionan fundiciones de las familias CM en todos los pesos, formas y tamaños ópticamente escalables.

### 2.5.1. Soporte para el castellano

Por José Luis Rivera <jlrm77@gmail.com>

Para posibilitar la silabación y cambiar todos los textos automáticos al castellano, use la orden:

```
\usepackage[spanish]{babel}
```

Como hay muchos acentos en castellano, debería usar

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

para poder meterlos con el teclado, y también

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

para que la silabación sea correcta.

Vea el cuadro 2.3 para un preámbulo adecuado para el castellano. Note que usamos la codificación de entrada latin1, que puede no ser correcta para su sistema.

La opción **spanish** añade algunos atajos (*shorthands*) útiles para la tipografía española del texto o las matemáticas. Estos atajos se explican en el cuadro 2.4.

En general, la opción **spanish** provee numerosos ajustes, pero el estilo está diseñado para que sea muy configurable. Para ello, se proporciona una serie de opciones de paquete, que en caso de emplearse deben ir *después* de **spanish**. Por ejemplo:

```
\usepackage[french,spanish,es-noindentfirst]{babel}
```

carga los estilos para el francés y el español, esta última como lengua principal; además, evita que **spanish** sangre el primer párrafo tras un título. Otras

Cuadro 2.4: Abreviaciones

'a 'e 'i 'o 'u	á é í ó ú <sup>a</sup>
'A 'E 'I 'O 'U	Á É Í Ó Ú <sup>a</sup>
'n 'N	ñ Ñ <sup>b</sup>
"u "U	"u "U
"i "I	"i "I
"a "A "o "O	Ordinales: 1ª, 1ªA, 1ªo, 1ªO
"er "ER	Ordinales: 1ªer, 1ªER
"c "C	"c "C
"rr "RR	rr, pero -r cuando se divide
"y	El antiguo signo para «y»
"-	Como \-, pero permite más divisiones
"=	Como -, pero permite mas divisiones <sup>c</sup>
"~	Guión estilístico <sup>d</sup>
"+ "+- "+--	Como -, -- y ---, pero sin división
~ ~- ~--	Lo mismo que el anterior.
" "	Permite mas divisiones antes y después <sup>e</sup>
"/	Una barra algo más baja
"	Divide un logotipo <sup>f</sup>
"< ">	"< ">
"‘ ”’	\begin{quoting} \end{quoting} <sup>g</sup>
<< >>	Lo mismo que el anterior.
? ‘ ! ‘	¿ ¡ <sup>h</sup>
"? "!	"? "!" alineados con la linea base <sup>i</sup>

<sup>a</sup> Requieren la opción `activeacute`. <sup>b</sup> La forma `~n` no está activada por omisión a partir de la versión 5. <sup>c</sup> `"=` viene a ser lo mismo que `"--"`. <sup>d</sup> Esta abreviación tiene un uso distinto en otras lenguas de babel. <sup>e</sup> Como en «entrada/salida». <sup>f</sup> Carece de uso en castellano. <sup>g</sup> Reemplazos para `<< o >>` con la opción `es-noquoting`. <sup>h</sup> No proporcionadas por este paquete, sino por cada tipo; figuran aquí como simple recordatorio. <sup>i</sup> útiles en rótulos en mayúsculas.

opciones se pueden ajustar por medio de macros, en particular aquellas que se puede desear cambiar en medio del documento (por ejemplo, el formato de la fecha). Las macros pueden incluirse en el archivo de configuración `spanish.cfg` para hacer cambios globales en un sistema completo.

El estilo modifica por omisión el diseño del texto o del documento en partes sustantivas. A continuación se enumeran los ajustes hechos al formato y las opciones y macros que los controlan.

- Todo el espacio es uniforme, con `\frenchspacing`.
- Se añade un punto después del número de todas las secciones. Se inhibe con la opción `es-nosectiondot`.
- Todos los párrafos incluyen un `\indentfirst`. Se inhibe con la opción `es-noindentfirst`

- Los entornos `enumerate` e `itemize` se adaptan a reglas castellanas.  
Las opciones `es-noenumerate` y `es-noitemize` inhiben estas modificaciones por separado, y la opción `es-nolists` inhibe ambas.  
Las macros `\spanishdashitems` y `\spanishsignitems` cambian los valores de las listas itemizadas a series de guiones o una serie alternativa de símbolos.
- Los contadores `\alph` y `\Alph` incluyen  $\tilde{n}$  después de  $n$ .
- Los marcadores de notas no numéricos se vuelven series de asteriscos.  
La opción `es-nolayout` inhabilita los cambios al formato del documento. Estos cambios afectan estas enumeraciones y llamadas a notas a pie de página.  
La opción `es-uppernames` hace versiones con mayúsculas para las traducciones de los encabezados (capítulo, bibliografía, etc.).  
La opción `es-tabla` reemplaza “cuadro” con “tabla”.  
La macro `\spanishcaption{}` cambia el valor de la palabra clave automáticamente. Por ejemplo, `\spanishcontents{Contenido}`.

Hay otras modificaciones que afectan la composición del texto, los caracteres activos y los atajos.

- Las comillas tipográficas en la codificación OT1 se toman de la fundición `lasy` en lugar de las macros `\ll` y `\gg`.
- *En modo matemático*, un punto seguido de un dígito escribe una coma decimal.  
La macro `\decimalpoint` restaura el decimal a punto, y la macro `\spanishdecimal{character}` asigna un carácter cualquiera.
- Se define un entorno `quoting` y dos abreviaturas `<<` y `>>` para formatear citas largas.  
La opción `es-noquoting` inhabilita los atajos `<<` y `>>` para el entorno `quoting`, pero se conservan los atajos `"'` y `"'`.  
La macro `\deactivatequoting` desactiva los atajos `<<` y `>>` temporalmente para habilitar los signos `<` and `>` en comparaciones numéricas y algunas macros de AMST<sub>E</sub>X.  
La macro `\spanishdeactivate{caracteres}` inhabilita temporalmente los atajos definidos por los caracteres incluidos en su argumento. Son elegibles los caracteres `.'~<>`.  
La opción `es-tilden` restaura el atajo `~` para escribir eñes. Sólo se provee para componer documentos viejos.

La opción `es-noshorthands` inhabilita todos los atajos activados por `"`, `'`, `<`, `>`, `~` y `.`

- Los ordinales castellanos se forman con la orden `\sptext` como en `1\sptext{er}`. El punto preceptuado está incluido automáticamente.
- Funciones matemáticas acentuadas (lím, máx, mín, mód) y espaciadas (arc cos, etc.).

Las órdenes `\unspacedoperators` y `\unaccentedoperators` inhabilitan estas funciones.

La macro `\spanishoperators{operators}` define los nombres de las funciones y operadores del castellano. Por ejemplo, la orden

```
\renewcommand{\spanishoperators}{arc\,ctg m\acute{i}n}
```

crea macros para estas funciones. Dentro de esta orden la macro `\`, añade espacios finos (en `\arcctg` en este caso), y la macro `\acute{letter}` añade un acento (como `m\acute{i}n` define `\min` (mín). No es necesario añadir la `\dotlessi` explícitamente.

- Se provee una orden `\dotlessi` para uso en modo matemático.
- Se añade un espacio fino al signo porcentual `\%`. La macro `\spanishplainpercent` lo inhibe localmente.
- Se provee una orden `\lsc` para producir versalitas minúsculas, para siglas o números romanos.
- Se redefine la orden `\roman` para escribir números romanos en versalitas en lugar de minúsculas.

La opción `es-preindex` llama automáticamente al paquete `romanidx.sty` para reparar llamadas de `makeindex` formateadas en versalitas. La macro `\spanishindexchars` define los caracteres que determinan las entradas de índice. Por omisión se usa `\spanishindexchars{|}{(}{)}`.

La opción `es-ucroman` convierte todos los numerales romanos en mayúsculas en lugar de versalitas, y la opción `es-lcroman` los convierte otra vez en minúsculas, si la macro `lsc` falla por algún motivo. La primera opción es preferible a la segunda, que es de hecho una falta ortográfica.

Tres macros controlan las mismas modificaciones temporalmente: `\spanishscroman`, `\spanishucroman`, and `\spanishlcroman`.

Algunas macros prestan facilidades adicionales para el formato de algunos documentos.

Opciones complejas	es-minimal	es-sloppy	es-noshorthands
<code>es-noindentfirst</code>	o	o	
<code>es-nosectiondot</code>	o	o	
<code>es-nolists</code>	o	o	
<code>es-noquoting</code>	o	o	o
<code>es-notilde</code>	o	o	o
<code>es-nodecimaldot</code>	o	o	o
<code>es-nolayout</code>		o	
<code>es-ucroman</code>	o		
<code>es-lcroman</code>		o	

Cuadro 2.5: Opciones globales del castellano

- Las macros `\spanishdatedel` y `\spanishdatede` controlan el formato del artículo en las fechas (`del` or `de`).
- La macro `\spanishreverseddate` pone el formato de fecha en la forma “Mes Día del Año”.
- La macro `\Today` inicia los nombres de los meses en mayúscula.
- Los puntos suspensivos en medio de una oración se escribe `\dots`

Finalmente, hay opciones que abrevian varias opciones al mismo tiempo. Se abrevian en el cuadro 2.5.

Finalmente, toda esta maquinaria permite construir opciones regionales del castellano. Las primeras de ellas son `mexico` y `mexico-com`. Ambas opciones redefinen las comillas del entorno `quoting`, y la primera añade `es-nodecimaldot`, como es costumbre en México y otros países de Centroamérica y el Caribe.

### Ajuste a la tipografía española

Es posible dar aspecto “español” a un texto compuesto en otro idioma “importando” el formato del texto definido en `spanish`. Basta cargar `spanish` como idioma principal, añadir a los extras del idioma seleccionado (esperanto en este caso) las características que se quieren importar, y seleccionar el nuevo idioma principal al principio del documento.

```
\usepackage[esperanto,spanish]{babel}
\makeatletter
\addto\extrasesperanto{\textspanish}
\declare@shorthand{esperanto}{^a}{\textormath{\es@sptext{a}}{\ensuremath{^a}}}
\declare@shorthand{esperanto}{^A}{\textormath{\es@sptext{A}}{\ensuremath{^A}}}
\makeatother
\AtBeginDocument{\selectlanguage{esperanto}}
```

De esta forma es posible componer texto en esperanto (u otro idioma cualquiera) y darle aspecto de “compuesto en España”.

### 2.5.2. La opción Unicode

Por Axel Kielhorn <A.Kielhorn@web.de>

Unicode es el modo más adecuado para incluir varios idiomas en un único documento, sobre todo cuando estos idiomas no utilizan el alfabeto latino. Hay dos motores  $\text{\TeX}$  que son capaces de procesar la entrada de texto Unicode:

$\text{\XeTeX}$  fue desarrollado para MacOS X pero está disponible para todas las arquitecturas. Se incluyó por primera vez en  $\text{\TeX}$ Live 2007.

$\text{\LuaTeX}$  es el sucesor de  $\text{\pdfTeX}$ . Se incluyó por primera vez en  $\text{\TeX}$ Live 2008.

A continuación se describe cómo se distribuye  $\text{\XeLaTeX}$  con  $\text{\TeX}$ Live 2010.

#### Inicio rápido

Para convertir un fichero  $\text{\LaTeX}$  existente a  $\text{\XeLaTeX}$  es necesario hacer lo siguiente:

1. Guarde el fichero como UTF-8
2. Elimine

```
\usepackage{inputenc}
\usepackage{fontenc}
\usepackage{textcomp}
```

del preámbulo.

3. Cambie

```
\usepackage[languageA]{babel}
```

por

```
\usepackage{polyglossia}
\setdefaultlanguage[babelshorthands]{languageA}
```



## 4. Añada

```
\usepackage[Ligatures=TeX]{fontspec}
```

al preámbulo.

El paquete `polyglossia`[\[19\]](#) es un sustituto de `babel`. Se encarga de los patrones de separación silábica y de la generación automática de cadenas de texto. La opción `babelshorthands` habilita la compatibilidad de abreviaturas de `babel` para alemán y catalán.

El paquete `fontspec`[\[21\]](#) se encarga de la carga de fuentes para  $\text{X}\text{\LaTeX}$  y  $\text{Lua}\text{\TeX}$ . La fuente predeterminada es Latin Modern Roman. Es un hecho poco conocido que algunos comandos  $\text{T}\text{\E}\text{\X}$  son ligaduras definidas in las fuentes Computer Modern. Si desea utilizarlas con una fuente no- $\text{T}\text{\E}\text{\X}$  debería simularlas. La opción `Ligatures=TeX` define las siguientes ligaduras:

--	—
---	---
,,	”
‘‘	“
!‘	¡
?‘	¿
,,	”
<<	«
>>	»

### Todo es *Γρηγο* para mí

Hasta ahora no se ha visto ninguna ventaja al usar un motor  $\text{T}\text{\E}\text{\X}$  Unicode. Esto cambia cuando abandonamos el alfabeto latino y nos movemos a un idioma más interesante como el griego o el ruso. Con un sistema basado en Unicode se puede simplemente<sup>5</sup> escribir los caracteres en el editor y  $\text{T}\text{\E}\text{\X}$  los entenderá.

Escribir en diferentes idiomas es fácil, basta con especificar los idiomas

---

<sup>5</sup>Depreciando el concepto de simple.

en el preámbulo.

```
\setdefaultlanguage{spanish}
\setotherlanguage[babelshorthands]{german}
```

Para escribir un párrafo en alemán puede usar el entorno alemán:

```
Texto en español.
\begin{german}
Deutscher Text.
\end{german}
Más texto en español.
```

Si sólo necesita utilizar una palabra en otro idioma puede usar el comando `\textlanguage`:

```
Texto en español. \textgerman{Gesundheit} es en realidad una palabra alemana.
```

Esto puede parecer innecesario ya que la única ventaja es una división correcta de palabras, pero cuando el segundo idioma es algo más exótico merece la pena el esfuerzo.

A veces, la fuente usada en el documento principal no contiene glifos que son necesarios en el idioma secundario<sup>6</sup>. La solución consiste en definir la fuente que se utilizará para este idioma. Cada vez que se activa un nuevo idioma, `polyglossia` comenzará comprobando si se ha definido una fuente para este idioma.

```
\newfontfamily\russianfont[Script=Cyrillic,(...)]{(font)}
```

Ahora usted puede escribir

```
\textrussian{Pravda} es un periódico ruso.
```

Como este documento está escrito con pdfL<sup>A</sup>T<sub>E</sub>X, no puedo mostrar los caracteres cirílicos reales.

El paquete `xgreek`[22] ofrece soporte para la escritura de griego antiguo o moderno (monotónico o politónico).

### Idiomas de escritura de Derecha a Izquierda (RTL).

Algunos idiomas se escriben de izquierda a derecha, otros se escriben de derecha a izquierda (RTL). `polyglossia` necesita el paquete `bidi`[23]<sup>7</sup> para dar soporte a idiomas RTL. El paquete `bidi` ha de ser el último paquete cargado,

<sup>6</sup>Latin Modern no contiene caracteres cirílicos.

<sup>7</sup>`bidi` no soporta Lua<sub>T</sub><sub>E</sub>X.

incluso después de `hyperref` que suele ser el último paquete. (Como `polyglossia` carga `bidi`, `polyglossia` ha de ser el último paquete cargado.)

El paquete `xepersian`[24] ofrece soporte para el persa. Proporciona comandos  $\text{\LaTeX}$  persas que le permiten introducir comandos como `\section` en persa, que lo hace muy atractivo para los hablantes nativos. `xepersian` es el único paquete que soporta kashida con  $\text{Xe}\text{\LaTeX}$ . Actualmente se está desarrollando un paquete para el siríaco que utiliza un algoritmo similar.

La fuente `IranNastaliq` proporcionada por el SCICT<sup>8</sup> está disponible en su sitio web <http://www.scict.ir/Portal/Home/Default.aspx>.

El paquete `arabxetex`[20] es compatible con varios idiomas de escritura árabe:

- árabe
- persa
- urdu
- sindhi
- pashto
- otomano (turco)
- kurdo
- kashmiri
- malayo (jawi)
- uighur

Proporciona una asignación de fuentes que habilita a  $\text{Xe}\text{\LaTeX}$  para procesar la entrada de caracteres usando la transcripción ASCII `Arab\TeX`.

IRMUG<sup>9</sup> proporciona fuentes que soportan algunos idiomas árabes en [http://wiki.irmug.org/index.php/X\\_Series\\_2](http://wiki.irmug.org/index.php/X_Series_2).

No hay paquetes disponibles para el hebreo porque no son necesarios. El soporte para hebreo de `polyglossia` debería ser suficiente. Pero es necesaria una fuente adecuada con verdadero Unicode hebreo. SBL hebreo es gratuito para uso no-comercial y está disponible en <http://www.sbl-site.org/educational/biblicalfonts.aspx>. Otra fuente disponible bajo licencia Open Font es Ezra SIL, disponible en [http://www.sil.org/computing/catalog/show\\_software.asp?id=76](http://www.sil.org/computing/catalog/show_software.asp?id=76).

---

<sup>8</sup>Supreme Council of Information and Communication Technology

<sup>9</sup>Iranian Mac User Group

Recuerde seleccionar la secuencia de comandos correcta:

```
\newfontfamily\hebrewfont[Script=Hebrew]{SBL Hebrew}
\newfontfamily\hebrewfont[Script=Hebrew]{Ezra SIL}
```

## Chino, japonés y coreano (CJK)

El paquete `xeCJK`<sup>[25]</sup> se encarga de la selección de fuentes y de la puntuación de estos idiomas.

## 2.6. El espacio entre palabras

Para conseguir un margen derecho recto en la salida,  $\text{\LaTeX}$  inserta cantidades variables de espacio entre las palabras. En tipografía inglesa, se inserta algo más de espacio al final de la oración, pues así el texto es más legible.  $\text{\LaTeX}$  supone que las oraciones terminan en puntos, signos de interrogación o signos de exclamación. Si un punto sigue una letra mayúscula, no se considera un final de oración, pues los puntos tras letras mayúsculas suelen indicar una abreviatura.

Cualquier excepción a esas premisas tiene que indicarla el autor. Una antibarra ante un espacio genera un espacio que no será expandido. Una tilde ‘~’ genera un espacio que no será expandido y además impide el salto de línea. La orden `\@` ante un punto indica que dicho punto termina una oración aunque siga a una letra mayúscula.

El Sr.~Aranda se alegró\  
cf.~Fig.~5\  
Adoro el LISP\@. ¿Y usted?

El Sr. Aranda se alegró  
cf. Fig. 5  
Adoro el LISP. ¿Y usted?

Al escribir en español, no se añade el espacio adicional tras los puntos. En inglés tal adición se puede desactivar con la orden

```
\frenchspacing
```

que manda a  $\text{\LaTeX}$  no insertar más espacio tras un punto que tras un signo ordinario. Es lo habitual en idiomas distintos del inglés, salvo en bibliografías. En tal caso, la orden `\@` no es necesaria.

## 2.7. Títulos, capítulos y secciones

Para ayudar al lector a orientarse en su libro, debería dividirlo en capítulos, secciones y subsecciones.  $\text{\LaTeX}$  lo permite mediante órdenes especiales que

toman el título de la sección como argumento. Es tarea suya el usarlos en el orden correcto.

Las siguientes órdenes de sección están disponibles para la clase `article`:

```
\section{...}  
\subsection{...}  
\subsubsection{...}  
\paragraph{...}  
\subparagraph{...}
```

Si quiere dividir su documento en partes **sin influir en la numeración de secciones o capítulos** puede usar

```
\part{...}
```

Cuando trabaje con las clases `report` o `book`, estará disponible una orden de sección adicional

```
\chapter{...}
```

Como la clase `article` no entiende de capítulos, es muy fácil añadir artículos como capítulos a un libro. El espacio entre secciones, la numeración y el tamaño de fundición de los títulos quedará correctamente establecido por  $\text{\LaTeX}$ .

Dos órdenes de sección son algo especiales:

- La orden `\part` no modifica la secuencia de numeración de los capítulos.
- La orden `\appendix` no toma ningún argumento. Solamente cambia la numeración de capítulos de números a letras.<sup>10</sup>

$\text{\LaTeX}$  crea un índice general tomando los encabezados de sección y los números de página del último ciclo de compilación del documento. La orden

```
\tableofcontents
```

sitúa el índice general en el lugar en que se ejecuta la orden. Un documento nuevo debe compilarse (`"\LaTeXarse"`) dos veces para conseguir un índice general correcto. A veces puede requerirse una tercera compilación.  $\text{\LaTeX}$  le dirá cuándo es necesario.

Todas las órdenes de sección listadas anteriormente tienen una versión “estrella”. Se trata de órdenes con el mismo nombre pero seguido de un asterisco `*`. Generan encabezados de sección que no aparecen en el índice

---

<sup>10</sup>Para el estilo artículo cambia la numeración de las secciones.

general y que no se numeran. La orden `\section{Ayuda}`, por ejemplo, tendría una versión estrella así: `\section*{Ayuda}`.

Normalmente los encabezados aparecen en el índice general exactamente como se introducen en el texto. A veces no es posible, porque el encabezado es demasiado largo y no cabe en el índice general. La entrada para el índice general puede indicarse como un argumento opcional antes del encabezado real.

```
\chapter[Título para el índice general]{Un largo
    y aburrido título que aparecerá en el texto}
```

El título de todo el documento se genera con la orden

```
\maketitle
```

El contenido del título tiene que definirse mediante las órdenes

```
\title{...}, \author{...} y opcionalmente \date{...}
```

antes de llamar a `\maketitle`. En el argumento de `\author`, puede poner varios nombres separados por órdenes `\and`.

Un ejemplo de algunas de las órdenes mencionadas arriba puede verse en la Figura 1.2 de la página 8.

Además de las órdenes de sección ya explicadas,  $\text{\LaTeX} 2_{\epsilon}$  tiene tres órdenes adicionales para usar con la clase `book`. Son útiles para dividir la publicación. Las órdenes alteran los encabezados de los capítulos y los números de página para que aparezcan como se ve en muchos libros (sobre todo ingleses):

`\frontmatter` debería ser la primerísima orden tras el comienzo del cuerpo del documento (`\begin{document}`). Cambia la numeración de páginas a números romanos y las secciones no estarán numeradas. Es como si usara las órdenes de sección con asterisco (p.ep. `\chapter*{Preface}`) pero las secciones aparecerán en el índice general.

`\mainmatter` viene justo antes del primer capítulo del libro. Activa los números de página arábigos y recomienza el contador de páginas.

`\appendix` marca el comienzo de material adicional en su libro. Tras esta orden los capítulos se numerarán con letras.

`\backmatter` debería insertarse antes de los últimos elementos del libro, como la bibliografía y el índice alfabético. No tiene efecto visual en las clases típicas.

## 2.8. Referencias cruzadas

En libros, informes y artículos, hay a menudo referencias cruzadas a figuras, cuadros y trozos especiales de texto.  $\text{\LaTeX}$  proporciona las siguientes órdenes para referenciar

$\text{\label{marcador}}$ ,  $\text{\ref{marcador}}$  y  $\text{\pageref{marcador}}$

donde *marcador* es un identificador escogido por el usuario.  $\text{\LaTeX}$  reemplaza  $\text{\ref}$  por el número de la sección, subsección, figura, tabla o teorema tras el que se sitúa la orden  $\text{\label}$  correspondiente.  $\text{\pageref}$  imprime el número de página de la página donde la orden  $\text{\label}$  se sitúa.<sup>11</sup> Como para los títulos de sección, se usan los números de la compilación previa.

Una referencia a esta subsección  $\text{\label{sec:esta}}$  aparece así:  
 “ver sección~ $\text{\ref{sec:esta}}$  en la página~ $\text{\pageref{sec:esta}}$ .”

Una referencia a esta subsección aparece así: “ver sección 2.8 en la página 40.”

## 2.9. Notas al pie

Con la orden

$\text{\footnote{texto al pie}}$

se imprime una nota al pie de la página actual. Deben ponerse las notas<sup>12</sup> tras la palabra u oración a la que se refieren. Las notas que se refieran a una sentencia o parte de ella deben por tanto ponerse tras la coma o el punto.<sup>13</sup>

Las notas al pie  $\text{\footnote{Esto es una nota al pie.}}$  se usan mucho en  $\text{\LaTeX}$ .

Las notas al pie<sup>a</sup> se usan mucho en  $\text{\LaTeX}$ .

<sup>a</sup>Esto es una nota al pie.

---

<sup>11</sup>Tenga en cuenta que estas órdenes no saben a qué cosa se refieren.  $\text{\label}$  solamente guarda el último número generado automáticamente.

<sup>12</sup>“nota” es una palabra polisémica.

<sup>13</sup>Fíjese en que las notas distraen al lector del flujo general del documento. Después de todo, todo el mundo lee las notas —somos una especie cotilla—, así que ¿por qué no integrar todo lo que quieres decir en el cuerpo del documento?<sup>14</sup>

<sup>14</sup>Una señal indicadora no se encuentra necesariamente en el sitio al que está señalando.

## 2.10. Palabras enfatizadas

Si un texto se escribe a máquina las palabras importantes **se enfatizan subrayándolas**.

```
\underline{texto}
```

En los libros impresos, sin embargo, las palabras se enfatizan componiéndolas con una fundición *cursiva*. L<sup>A</sup>T<sub>E</sub>X proporciona la orden

```
\emph{texto}
```

para enfatizar texto. Lo que hace realmente la orden con su argumento depende del contexto:

```
\emph{Si usa énfasis en un
fragmento de texto ya
enfatizado, entonces
\LaTeX{} usa la \fontnomo{}
\emph{normal} para
enfatizar.}
```

*Si usa énfasis en un fragmento de texto ya enfatizado, entonces L<sup>A</sup>T<sub>E</sub>X usa la fundición normal para enfatizar.*

Fíjese bien en la diferencia entre mandar a L<sup>A</sup>T<sub>E</sub>X que *enfatice* algo y mandarle que use una *fundición* diferente:

```
\textit{También puede
\emph{enfatar} texto
aunque esté en cursiva,}
\textsf{en \fontnomo{}
\emph{sin serifado},}
\texttt{o en estilo
\emph{mecanográfico}.}
```

*También puede enfatizar texto aunque esté en cursiva, en fundición sin serifado, o en estilo mecanográfico.*

## 2.11. Entornos

```
\begin{entorno} texto \end{entorno}
```

Aquí *entorno* es un nombre de entorno. Los entornos pueden anidarse uno dentro de otro mientras se mantenga el orden correcto.

```
\begin{aaa}...\begin{bbb}...\end{bbb}...\end{aaa}
```

En las siguientes secciones se explican todos los entornos importantes.



### 2.11.1. Listas (`itemize`, `enumerate` y `description`)

El entorno `itemize` es adecuado para listas simples, el entorno `enumerate` para listas enumeradas y el entorno `description` para descripciones.

```
\flushleft
\begin{enumerate}
\item Puede mezclar los
entornos de lista a su gusto:
\begin{itemize}
\item Pero podría empezar a
parecer estúpido.
\item[-] Con un guión.
\end{itemize}
\item Así que recuerde:
\begin{description}
\item[Estupideces] no mejoran
por ponerlas en una lista.
\item[Lucideces] sin embargo,
pueden parecer hermosas en
una lista.
\end{description}
\end{enumerate}
```

1. Puede mezclar los entornos de lista a su gusto:

- Pero podría empezar a parecer estúpido.
- Con un guión.

2. Así que recuerde:

**Estupideces** no mejoran por ponerlas en una lista.

**Lucideces** sin embargo, pueden parecer hermosas en una lista.

### 2.11.2. Alineación (`flushleft`, `flushright` y `center`)

Los entornos `flushleft` y `flushright` generan párrafos alineados a la izquierda o a la derecha respectivamente. El entorno `center` genera texto centrado. Si no indica los saltos de línea mediante `\\`, `LATEX` los determinará automáticamente.

```
\begin{flushleft}
Este texto está alineado a
la izquierda. \LaTeX{} no trata
de justificar las líneas, así
que así quedan.
\end{flushleft}
```

Este texto está alineado a la izquierda.  
`LATEX` no trata de justificar las líneas, así que así quedan.

```
\begin{flushright}
Texto alineado\\a la derecha.
\LaTeX{} no trata de
justificar las líneas.
\end{flushright}
```

Texto alineado  
a la derecha. `LATEX` no trata de justificar  
las líneas.

```
\begin{center}
En el centro\\de la Tierra
\end{center}
```

En el centro  
de la Tierra

### 2.11.3. Citas (`quote`, `quotation` y `verse`)

El entorno `quote` es útil para citas, frases importantes y ejemplos.

Una regla empírica tipográfica para la longitud de renglón es:  
`\begin{quote}`  
 En promedio, ningún renglón debería tener más de 66 signos.  
`\end{quote}`  
 Por ello las páginas de `\LaTeX{}` tienen márgenes tan anchos por omisión, y los periódicos usan múltiples columnas.

Una regla empírica tipográfica para la longitud de renglón es:

En promedio, ningún renglón debería tener más de 66 signos.

Por ello las páginas de `\LaTeX{}` tienen márgenes tan anchos por omisión, y los periódicos usan múltiples columnas.

Hay dos entornos similares: el `quotation` y el `verse`. El entorno `quotation` es útil para citas largas que se extienden varios párrafos, porque sangra la primera línea de cada párrafo. El entorno `verse` es útil para poemas donde son importantes los saltos de línea. Los renglones se separan mediante `\\` al final de línea y las estrofas mediante un renglón vacío.

He aquí un fragmento de todo un monstruo: Quevedo.  
`\begin{flushleft}`  
`\begin{verse}`  
 Pasa veloz del mundo la figura,\\  
 y la muerte los pasos apresura;\\  
 la vida nunca para,\\  
 ni el Tiempo vuelve atrás la anciana cara.  
`\end{verse}`  
`\end{flushleft}`

He aquí un fragmento de todo un monstruo: Quevedo.

Pasa veloz del mundo la figura,  
 y la muerte los pasos apresura;  
 la vida nunca para,  
 ni el Tiempo vuelve atrás la anciana cara.

### 2.11.4. Resumen (`abstract`)

En publicaciones científicas es habitual empezar con un resumen que da al lector una idea rápida de lo que puede esperar. `\LaTeX{}` proporciona el entorno `abstract` con este propósito. Normalmente `abstract` se usa para documentos compuestos con la clase `article`.

`\begin{abstract}`  
 Esta frase está en el resumen, es un 80% del ancho total.  
`\end{abstract}`  
 Esta frase está fuera del resumen, así que es más ancha.

Esta frase está en el resumen, es un 80% del ancho total.

Esta frase está fuera del resumen, así que es más ancha.

### 2.11.5. Citas literales (`verbatim`)

El texto encerrado entre `\begin{verbatim}` y `\end{verbatim}` se escribirá directamente, como escrito a máquina, con todos los saltos de línea y espacios, sin ejecutar ninguna orden  $\text{\LaTeX}$ .

Dentro de un párrafo, un comportamiento similar se puede obtener con

`\verb+texto+`

El signo `+` puede sustituirse por cualquier otro, salvo por letras, `*` por espacios; sirve meramente para delimitar. Muchos ejemplos de  $\text{\LaTeX}$  en esta introducción se componen mediante esta orden.

Con `\verb|\u{u}|` obtengo `\u{u}`.

```
\begin{verbatim}
(LOOP
  (PRINT "HOLA MUNDO\n"))
\end{verbatim}
```

Con `\u{u}` obtengo ũ.

```
(LOOP
  (PRINT "HOLA MUNDO\n"))
```

```
\begin{verbatim*}
la versión con asterisco
del entorno verbatim
destaca los espacios (no
finales) del texto
\end{verbatim*}
```

```
la versión con asterisco
del entorno verbatim
destaca los espacios
finales) del texto
```

La orden `\verb` puede usarse también con un asterisco:

```
\verb*|tal que así :-)|
```

```
tal que así :-)
```

El entorno `verbatim` y la orden `\verb` pueden estar prohibidos dentro de los parámetros de algunas órdenes.

### 2.11.6. Tablas (`tabular`)

El entorno `tabular` se usa para componer lindas tablas con líneas opcionales horizontales o verticales.  $\text{\LaTeX}$  determina el ancho de las columnas automáticamente.

El argumento *espec* de la orden

`\begin{tabular}[pos]{espec}`

define el formato de la tabla. Use un `l` para una columna de texto alineado por la izquierda, `r` para alineación por la derecha y `c` para texto

centrado; `p{anchura}` para una columna con texto justificado con saltos de renglón y `|` para una línea vertical.

Si el texto de una columna es demasiado ancha para la página,  $\text{\LaTeX}$  no lo partirá automáticamente. Mediante `p{anchura}` puede definir un tipo de columna especial que partirá el texto como en un párrafo normal.

El argumento *pos* indica la posición vertical de la tabla relativa a la base del texto alrededor. Use una de las letras `t`, `b` o `c` para indicar alineación por lo alto, por lo bajo o por el centro, respectivamente.

En un entorno `tabular`, `&` salta a la columna siguiente, `\` comienza un nuevo renglón y `\hline` inserta una línea horizontal. Puede añadir líneas parciales usando `\cline{j-i}`, donde *j* e *i* son los números de las columnas sobre las que debería extenderse la línea.

```
\begin{tabular}{|r|l|}
\hline
7C0 & hexadecimal \\
3700 & octal \\
11111000000 & binario \\
\hline
1984 & decimal \\
1194 & docenal \\
\hline
\end{tabular}
```

7C0	hexadecimal
3700	octal
11111000000	binario
1984	decimal
1194	docenal

```
\begin{tabular}{|p{4.7cm}|}
\hline
Bienvenidos a mi párrafo.
Esperamos que se diviertan
con el espectáculo.\\
\hline
\end{tabular}
```

Bienvenidos a mi párrafo. Esperamos que se diviertan con el espectáculo.
--

El separador de columnas puede indicarse con el constructo `@{...}`. Esta orden elimina el espacio entre columnas y lo reemplaza con lo que se ponga entre las llaves. Un uso común de esta orden se explica abajo en un problema de alineación de decimales. Otra aplicación posible es suprimir el espacio adicional de una tabla mediante `@{}`.

```
\begin{tabular}{@{} l @{}}
\hline
sin espacio extra\\
\hline
\end{tabular}
```

sin espacio extra
-------------------

```
\begin{tabular}{l}
\hline
con espacio a izq. y dcha.\\
\hline
\end{tabular}
```

con espacio a izq. y dcha.
----------------------------

Puesto que no hay manera predefinida para alinear columnas de números por el decimal,<sup>15</sup> podemos “chapucear” y hacerlo mediante dos columnas: enteros alineados por la derecha y fracciones alineadas por la izquierda. La orden `@{'}` en el renglón `\begin{tabular}` reemplaza el espacio normal entre columnas por una comilla “'”, lo que da el aspecto de una sola columna alineada por una coma decimal. No olvide reemplazar el punto decimal en sus números por un separador de columnas (`&`). La cabecera de la “columna” puede conseguirse con la orden `\multicolumn`.

```
\begin{tabular}{c r @{'} l}
Expresión con pi & & \\
\multicolumn{2}{c}{Valor} \\
\hline
$\pi$ & 3&1416 & \\
$\pi^{\pi}$ & 36&46 & \\
$(\pi^{\pi})^{\pi}$ & 80662&7 & \\
\end{tabular}
```

Expresión con pi	Valor
$\pi$	3'1416
$\pi^{\pi}$	36'46
$(\pi^{\pi})^{\pi}$	80662'7

Aunque los signos recomendado y permitido por ISO para los decimales son una coma baja (,) o un punto bajo (.) respectivamente, este ejemplo usa el signo tradicional para el decimal en la tipografía española, que es una coma alta ('), y muestra que puede usarse un símbolo cualquiera para alinear con el marcador `@{ }`.

```
\begin{tabular}{|c|c|}
\hline
\multicolumn{2}{|c|}{Unu} \\
\hline
Du & Tri! \\
\hline
\end{tabular}
```

Unu	
Du	Tri!

El material compuesto con el entorno `tabular` siempre permanece junto en una misma página. Si quiere componer tablas largas, debe usar entornos `longtable`.

## 2.12. Elementos deslizantes

Actualmente la mayoría de las publicaciones contienen muchas figuras y cuadros. Estos elementos requieren un tratamiento especial, porque no

<sup>15</sup>Compruebe si tiene instalado en su sistema el paquete `dcolumn`.

pueden dividirse entre dos páginas. Un método posible sería empezar una nueva página cada vez que una figura o un cuadro es demasiado grande para encajar en la página actual. Este enfoque dejaría páginas parcialmente vacías, lo que da mal aspecto.

La solución a este problema es deslizar (*dejar flotar*) cualquier figura o cuadro que no encaje en la página actual hacia una página posterior, y rellenar la página actual con texto del documento. L<sup>A</sup>T<sub>E</sub>X ofrece dos entornos para elementos deslizantes: uno para cuadros y otro para figuras. Para aprovecharlos bien es importante entender aproximadamente cómo maneja L<sup>A</sup>T<sub>E</sub>X internamente los deslizantes. En caso contrario, pueden volverse una fuente de frustraciones, si L<sup>A</sup>T<sub>E</sub>X nunca los pone donde usted quiere que vayan.

Echemos primero un vistazo a las órdenes que L<sup>A</sup>T<sub>E</sub>X proporciona para deslizantes.

Cualquier cosa que vaya dentro de un entorno **figure** o **table** se tratará como deslizante. Ambos entornos admiten un parámetro opcional llamado *colocador*.

`\begin{figure}[colocador]` ó `\begin{table}[colocador]`

Este parámetro se usa para decir a L<sup>A</sup>T<sub>E</sub>X dónde se puede deslizar el elemento. Se contruye un *colocador* mediante una cadena de *permisos de deslizamiento*. Véase el cuadro 2.6.

P.ej. un cuadro podría empezar con el renglón siguiente:

`\begin{table}[!hbp]`

El colocador `[!hbp]` permite que L<sup>A</sup>T<sub>E</sub>X coloque el cuadro justo aquí (**h**) o abajo (**b**) en alguna página o en una página especial con deslizantes (**p**), todo ello incluso si no queda tan bien (**!**). Si no se indica un colocador, las clases típicas suponen `[tbp]`.

Cuadro 2.6: Permisos de deslizamiento.

Signo	Permiso para deslizar...
<b>h</b>	aquí ( <i>here</i> ) en el mismo lugar del texto donde aparece. Útil sobre todo para elementos pequeños.
<b>t</b>	arriba ( <i>top</i> ) en la página.
<b>b</b>	abajo ( <i>bottom</i> ) en la página.
<b>p</b>	en una <i>página</i> especial sólo con deslizantes.
<b>!</b>	sin considerar la mayoría de los parámetros internos <sup>a</sup> , que podrían impedir su colocación.

---

<sup>a</sup>Como el número máximo de deslizantes por página permitido.

L<sup>A</sup>T<sub>E</sub>X colocará todos los deslizantes que encuentre según el colocador indicado por el autor. Si un deslizante no puede colocarse en la página actual, quedará pospuesto en la cola de *figuras* o en la de *cuadros*.<sup>16</sup> Cuando comienza una nueva página, L<sup>A</sup>T<sub>E</sub>X comprueba antes si es posible rellenar un página especial de deslizantes, con deslizantes de la colas. Si no es posible, se considera el primer deslizante de cada cola como si acabase de aparecer en el texto: L<sup>A</sup>T<sub>E</sub>X intenta de nuevo colocarlo según su colocador (salvo por la ‘h’, que ya no es posible). Se sitúa cualquier deslizante nuevo que aparezca en el texto dentro de las colas apropiadas. L<sup>A</sup>T<sub>E</sub>X mantiene estrictamente el orden original de aparición para cada tipo de deslizante. Por eso una figura que no puede colocarse empuja todas las demás figuras hacia el final del documento. Por tanto:

Si L<sup>A</sup>T<sub>E</sub>X no coloca los deslizantes como usted esperaba, suele ser por culpa de un solo deslizante atascado en una de las dos colas.

Aunque se puede dar a L<sup>A</sup>T<sub>E</sub>X un colocador de una sola letra, causa problemas. Si el deslizante no encaja en el lugar indicado se queda atorado, y bloquea los deslizantes siguientes. En concreto, no debería nunca jamás usar la opción [h] —es tan mala que en versiones recientes de L<sup>A</sup>T<sub>E</sub>X se sustituye automáticamente por [ht]—.

Habiendo explicado lo difícil, quedan más cosas por mencionar sobre los entornos `table` y `figure`. Con la orden

```
\caption{texto del pie}
```

puede definir un pie para el deslizante. L<sup>A</sup>T<sub>E</sub>X añadirá un número correlativo y la cadena “Figura” o “Cuadro”.

Las dos órdenes

```
\listoffigures y \listoftables
```

funcionan análogamente a la orden `\tableofcontents`, imprimiendo un índice de figuras o cuadros, respectivamente. Tales índices muestran los pies completos, así que si tiende a usar pies largos debe tener una versión más corta del pie para los índices. Se consigue poniendo la versión corta entre corchetes tras la orden `\caption`.

```
\caption[Corto]{LLLLLLaaaaaaarrrrrrrrgggggggoooooo}
```

Con `\label` y `\ref`, puede crear una referencia al flotante dentro del texto.

---

<sup>16</sup>Son colas FIFO —‘first in first out’—: primero en entrar, primero en salir.

El ejemplo siguiente dibuja un cuadrado y lo inserta en el documento. Podría usarlo si quisiera reservar espacio para imágenes que vaya a pegar en el documento ya impreso.

```
La figura~\ref{blanco} es un ejemplo de Arte Pop.
\begin{figure}[!hbp]
\makebox[\textwidth]{\framebox[5cm]{\rule{0pt}{5cm}}}
\caption{Cinco por cinco centímetros.\label{blanco}}
\end{figure}
```

En el ejemplo de arriba, L<sup>A</sup>T<sub>E</sub>X tratará *con insistencia* (!) de colocar la figura *aquí* (h).<sup>17</sup> Si no es posible, trata de colocar la figura *abajo* (b). Si no puede colocar la figura en la página actual, determina si es posible crear una página de deslizantes que contenga esta figura y quizás algunos cuadros de la cola de cuadros. Si no hay bastante material para una página especial de deslizantes, L<sup>A</sup>T<sub>E</sub>X comienza una nueva página, y una vez más trata la figura como si acabara de aparecer en el texto.

En ciertas circunstancias podrá requerirse el uso de la orden

`\clearpage` o incluso de `\cleardoublepage`

Manda a L<sup>A</sup>T<sub>E</sub>X colocar inmediatamente todos los deslizantes que quedan en las colas y después empezar una página nueva. `\cleardoublepage` incluso salta a una nueva página a la derecha.

Aprenderá a incluir dibujos POSTSCRIPT en sus documentos L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> más tarde en esta introducción.

## 2.13. Protección de órdenes frágiles

El texto dado como argumento de órdenes como `\caption` o `\section` puede aparecer más de una vez en el documento (p.ej. en el índice además de en el cuerpo del documento). Algunas órdenes no funcionarán cuando se usen en el argumento de órdenes como `\section`. La compilación de su documento fracasará. Tales órdenes se llaman órdenes frágiles —por ejemplo, `\footnote` o `\phantom`. Estas órdenes frágiles necesitan protección. Puede protegerlas precediéndolas con la orden `\protect`.

`\protect` sólo se refiere a la orden que le sigue, ni siquiera a sus argumentos. En la mayoría de los casos un `\protect` superfluo no hará daño.

```
\section{Soy muy considerado
\protect\footnote{y protejo mis notas al pie.}}
```

---

<sup>17</sup>suponiendo que la cola de figuras está vacía.





## Capítulo 3

# Composición de fórmulas matemáticas

¡Ahora está listo! En este capítulo, abordaremos la mayor aptitud de  $\text{\TeX}$ : la composición matemática. Pero cuidado, este capítulo solo trata la superficie. Aunque lo que se explica aquí basta para mucha gente, no desespere si no encuentra aquí la solución a sus necesidades de composición matemática. Es muy probable que su problema haya sido abordado en  $\mathcal{A}\mathcal{M}\mathcal{S}\text{-}\text{\TeX}$ <sup>1</sup>

### 3.1. Generalidades

$\text{\LaTeX}$  tiene un modo especial para componer matemáticas. Hay dos posibilidades: escribir las matemáticas dentro de un párrafo, en el mismo renglón que el resto del texto, o partir el párrafo para componer las matemáticas aparte, destacadas. El texto matemático *dentro* del párrafo se introduce entre  $\backslash($  y  $\backslash)$ , entre  $\$$  y  $\$$ , o entre  $\backslash\text{begin}\{\text{math}\}$  y  $\backslash\text{end}\{\text{math}\}$ .

Sume  $\$a\$$  al cuadrado y  $\$b\$$  al cuadrado para obtener  $\$c\$$  al cuadrado. Más formalmente:  
 $\$c^{\wedge\{2\}}=a^{\wedge\{2\}}+b^{\wedge\{2\}}\$$

Sume  $a$  al cuadrado y  $b$  al cuadrado para obtener  $c$  al cuadrado. Más formalmente:  
 $c^2 = a^2 + b^2$

$\backslash\text{\TeX}\{\}$  se pronuncia como  
 $\backslash(\backslash\text{\tau}\backslash\text{\epsilon}\backslash\text{\chi}\backslash)\backslash\backslash[6\text{pt}]$   
 $100\text{-m}\$^{\wedge\{3\}}\$$  de agua $\backslash\backslash[6\text{pt}]$   
De todo  
 $\backslash\text{begin}\{\text{math}\}\backslash\text{heartsuit}\backslash\text{end}\{\text{math}\}$

$\text{\TeX}$  se pronuncia como  $\tau\epsilon\chi$ .  
 $100\text{ m}^3$  de agua  
De todo ♡

---

<sup>1</sup>La *American Mathematical Society* (Sociedad Matemática Estadounidense) ha producido una potente extensión de  $\text{\LaTeX}$ . Muchos de los ejemplos de este capítulo hacen uso de dicha extensión. Todas las distribuciones recientes de  $\text{\TeX}$  la proporcionan. Si la suya no la tiene, visite `macros/latex/required/amslatex`.

Si quiere que sus ecuaciones o fórmulas matemáticas más grandes se sitúen destacadas aparte del resto del párrafo, es preferible *aislarlas*. Para ello, puede encerrarlas entre `\[` y `\]`, entre `\begin{displaymath}` y `\end{displaymath}`, o entre `\begin{equation}` y `\end{equation}`.

Sume  $a$  al cuadrado y  $b$  al cuadrado para obtener  $c$  al cuadrado. Más formalmente:

```
\begin{displaymath}
c^2=a^2+b^2
\end{displaymath}
```

o puede teclear menos con:

```
\[c^2=a^2+b^2\]
```

Sume  $a$  al cuadrado y  $b$  al cuadrado para obtener  $c$  al cuadrado. Más formalmente:

$$c^2 = a^2 + b^2$$

o puede teclear menos con:

$$c^2 = a^2 + b^2$$

Si quiere que L<sup>A</sup>T<sub>E</sub>X enumere sus ecuaciones, puede usar el entorno `equation`. Puede etiquetar mediante `\label` la ecuación con un número y referirse a éste desde otro lugar del texto usando `\ref` o la orden `\eqref` del paquete `amsmath`:

```
\begin{equation} \label{eq:eps}
\epsilon > 0
\end{equation}
De (\ref{eq:eps}), se deduce
\ldots{} De \eqref{eq:eps}
se deduce lo mismo.
```

$$\epsilon > 0 \quad (3.1)$$

De (3.1), se deduce ... De (3.1) se deduce lo mismo.

Observe las diferencias de estilo entre las ecuaciones en párrafo y las aisladas:

```
 $\lim_{n \to \infty}
\sum_{k=1}^n \frac{1}{k^2}
= \frac{\pi^2}{6}$
```

$$\lim_{n \rightarrow \infty} \sum_{k=1}^n \frac{1}{k^2} = \frac{\pi^2}{6}$$

```
\begin{displaymath}
\lim_{n \to \infty}
\sum_{k=1}^n \frac{1}{k^2}
= \frac{\pi^2}{6}
\end{displaymath}
```

$$\lim_{n \rightarrow \infty} \sum_{k=1}^n \frac{1}{k^2} = \frac{\pi^2}{6}$$

Hay diferencias entre *modo matemático* y *modo texto*. Por ejemplo, en *modo matemático*:

1. La mayoría de los espacios y saltos de línea no significan nada, pues todos los espacios se deducen lógicamente de las expresiones matemáticas, o tienen que ser indicados con órdenes especiales como `\,`, `\quad` o `\qquad`.

2. No se permiten renglones vacíos. Sólo un párrafo por fórmula.
3. Cada letra se considera como nombre de una variable y como tal será compuesta. Si quiere componer texto normal dentro de una fórmula (tipo redondo y espaciado normal) entonces tiene que introducir el texto usando las órdenes `\textrm{...}` (véase también la sección 3.7 en la página 61).

```
\begin{equation}
\forall x \in \mathbf{R} :
\quad x^2 \geq 0
\end{equation}
```

$$\forall x \in \mathbf{R} : \quad x^2 \geq 0 \quad (3.2)$$

```
\begin{equation}
x^2 \geq 0 \quad \text{para todo } x \in \mathbf{R}
\end{equation}
```

$$x^2 \geq 0 \quad \text{para todo } x \in \mathbf{R} \quad (3.3)$$

Los matemáticos pueden ser muy quisquillosos sobre qué símbolos usar: sería tradicional usar aquí la ‘negrita de pizarra’, que se obtiene usando `\mathbb` del paquete `amssymb` o `amssymb`. El último ejemplo se convierte en

```
\begin{displaymath}
x^2 \geq 0 \quad \text{para todo } x \in \mathbb{R}
\end{displaymath}
```

$$x^2 \geq 0 \quad \text{para todo } x \in \mathbb{R}$$

## 3.2. Agrupación en modo matemático

La mayoría de las órdenes en modo matemático actúan sólo sobre el siguiente carácter, así que si quiere que una orden afecte a varios caracteres, debe agruparlos juntos entre llaves: `{...}`.

```
\begin{equation}
a^{x+y} \neq a^{x+y}
\end{equation}
```

$$a^x + y \neq a^{x+y} \quad (3.4)$$

## 3.3. Construcción de bloques de una fórmula matemática

Esta sección describe las órdenes más importantes usadas en composición matemática. Eche un vistazo a la sección 3.10 en la página 65 donde se muestra una lista detallada de órdenes para componer símbolos matemáticos.

Las **letras griegas minúsculas** se introducen con `\alpha`, `\beta`, `\gamma`, ..., las mayúsculas se introducen con `\Gamma`, `\Delta`, ...<sup>2</sup>

`\lambda, \xi, \pi, \mu, \Phi, \Omega`

$\lambda, \xi, \pi, \mu, \Phi, \Omega$

Los **exponentes y subíndices** pueden indicarse con los caracteres `^` y `_`.

`$a_{1}$ \quad $x^{2}$ \quad`  
`$e^{-\alpha t}$ \quad`  
`$a^{3}_{ij}$ \quad`  
`$e^{x^2} \neq e^{x^2}$`

$a_1 \quad x^2 \quad e^{-\alpha t} \quad a_{ij}^3$   
 $e^{x^2} \neq e^{x^2}$

La **raíz cuadrada** se introduce como `\sqrt`; la raíz  $n$ ésima se genera con `\sqrt[n]`. El tamaño del signo de la raíz lo determina automáticamente L<sup>A</sup>T<sub>E</sub>X. Si sólo necesita el signo (habitual en la tradición anglosajona, pero no en la tipografía española), use `\surd`.

`$\sqrt{x}$ \quad`  
`$\sqrt{x^2 + \sqrt{y}}$ \quad`  
`\quad $\sqrt[3]{2}$ \quad [3pt]`  
`$\surd[x^2 + y^2]$`

$\sqrt{x} \quad \sqrt{x^2 + \sqrt{y}} \quad \sqrt[3]{2}$   
 $\sqrt{x^2 + y^2}$

`$\Psi = v_1 \cdot v_2`  
`\quad \ldots \quad`  
`$n! = 1 \cdot 2`  
`\quad \ldots (n-1) \cdot n$`

$\Psi = v_1 \cdot v_2 \dots \quad n! = 1 \cdot 2 \dots (n-1) \cdot n$

Las órdenes `\overline` y `\underline` crean **líneas horizontales** justo encima o debajo de una expresión.

`$\overline{m+n}$`

$\overline{m+n}$

Las órdenes `\overbrace` y `\underbrace` crean **llaves horizontales** largas sobre o bajo una expresión.

`$\underbrace{a+b+\dots+z}_{26}$`

$\underbrace{a+b+\dots+z}_{26}$

Para añadir acentos matemáticos como flechas pequeñas o tildes a las variables, puede usar las órdenes dadas en el Cuadro 3.1 de la página 65. Se

<sup>2</sup>No hay definida una alfa mayúscula en L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> porque parece igual que una A latina normal. Cuando se termine el nuevo código matemático, las cosas cambiarán.

consiguen circunflejos anchos y tildes que cubren varios caracteres mediante `\widetilde` y `\widehat`. El símbolo `'` produce una prima.

```
\begin{displaymath}
y=x^2\quad y'=2x\quad y''=2
\end{displaymath}
```

$$y = x^2 \quad y' = 2x \quad y'' = 2$$

Los **vectores** suelen indicarse añadiendo flechas pequeñas encima de una variable. Esto se hace con la orden `\vec`. Las dos órdenes `\overrightarrow` y `\overleftarrow` son útiles para denotar un vector desde  $A$  hasta  $B$ .

```
\begin{displaymath}
\vec a\quad\overrightarrow{AB}
\end{displaymath}
```

$$\vec{a} \quad \overrightarrow{AB}$$

No se suele escribir un punto explícito para indicar una multiplicación; sin embargo, a veces sí se escribe para ayudar a los ojos del lector a agrupar los elementos de una fórmula. Puede usar `\cdot` en estos casos:

```
\begin{displaymath}
v = {\sigma}_1 \cdot {\sigma}_2 \tau_1 \cdot \tau_2
\end{displaymath}
```

$$v = \sigma_1 \cdot \sigma_2 \tau_1 \cdot \tau_2$$

Los nombres de funciones como log suelen componerse en una fundición redonda, y no en cursiva como se hace con las variables, así que L<sup>A</sup>T<sub>E</sub>X proporciona las siguientes órdenes para componer las nombres de funciones más importantes, tanto para documentos en inglés...

```
\arccos \cos \csc \exp \ker \limsup
\arcsin \cosh \deg \gcd \lg \ln
\arctan \cot \det \hom \lim \log
\arg \coth \dim \inf \liminf \max
\sinh \sup \tan \tanh \min \Pr
\sec \sin
```

...como para documentos en español:

```
\cosec \arcsen \deg \arctg \cotg \sen
\arg \inf \senh \tg \tgh
```

```
\[\lim_{x \rightarrow 0}
\frac{\sen x}{x}=1\]
```

$$\lim_{x \rightarrow 0} \frac{\sen x}{x} = 1$$

Para la función módulo, hay dos órdenes: `\bmod` para el operador binario “ $a$  mód  $b$ ” y `\pmod` para expresiones tales como “ $x \equiv a$  (mód  $b$ ).”

```
$a\bmod b$\$
$x\equiv a \pmod{b}$
```

$$a \text{ mód } b$$

$$x \equiv a \text{ (mód } b)$$

Una **fracción** vertical se compone con la orden `\frac{...}{...}`. A menudo es preferible la forma horizontal  $1/2$ , porque queda mejor para cantidades pequeñas de “material fraccional”.

```
$1\frac{1}{2}$~horas
\begin{displaymath}
\frac{x^2}{k+1}\quad
x^{\frac{2}{k+1}}\quad
x^{1/2}
\end{displaymath}
```

$1\frac{1}{2}$  horas

$$\frac{x^2}{k+1} \quad x^{\frac{2}{k+1}} \quad x^{1/2}$$

Para componer coeficientes binomiales o estructuras similares, puede usar la orden `\binom` del paquete `amsmath`.

```
\begin{displaymath}
\binom{n}{k}\quad\mathrm{C}_n^k
\end{displaymath}
```

$$\binom{n}{k} \quad C_n^k$$

Para relaciones binarias puede ser útil apilar símbolos uno sobre otro. `\stackrel` pone el símbolo dado en el primer argumento con tamaño superíndice sobre el segundo, que se coloca en su posición habitual.

```
\begin{displaymath}
\int f_N(x) \stackrel{!}{=} 1
\end{displaymath}
```

$$\int f_N(x) \stackrel{!}{=} 1$$

El operador **integral** se genera con `\int`, el **sumatorio** con `\sum` y el **productorio** con `\prod`. Los límites superior e inferior se indican con `^` y `_` como los superíndices y subíndices.<sup>3</sup>

```
\begin{displaymath}
\sum_{i=1}^n \quad \int_0^{\frac{\pi}{2}} \quad \prod_{\epsilon}
\end{displaymath}
```

$$\sum_{i=1}^n \quad \int_0^{\frac{\pi}{2}} \quad \prod_{\epsilon}$$

Para controlar más aún la colocación de índices en expresiones complejas, `amsmath` proporciona dos herramientas adicionales: la orden `\substack` y el entorno `subarray`:

---

<sup>3</sup> `AMS- $\LaTeX$`  además tiene super-/subíndices multi-renglón.

$$\sum_{\substack{0 \leq i < n \\ 1 \leq j < m}} P(i,j) = \sum_{\begin{subarray}{l} i \in I \\ 1 \leq j < m \end{subarray}} Q(i,j)$$

$$\sum_{\substack{0 \leq i < n \\ 1 \leq j \leq m}} P(i, j) = \sum_{\substack{i \in I \\ 1 \leq j \leq m}} Q(i, j)$$

TEX proporciona todo tipo de símbolos como **llaves** y otros delimitadores (p.ej.  $\langle \parallel \updownarrow \rangle$ ). Paréntesis y corchetes pueden introducirse con las teclas correspondientes, y llaves con `\{`, pero el resto de los delimitadores se generan con órdenes especiales (p.ej. `\updownarrow`). Para una lista de todos los delimitadores disponibles, vea el Cuadro 3.7 en la página 67.

$$\{a,b,c\}\neq\{a,b,c\}$$

$$a, b, c \neq \{a, b, c\}$$

Si pone la orden `\left` ante un delimitador de apertura, y `\right` ante un delimitador de cierre, TeX determinará automáticamente el tamaño correcto del delimitador. Tenga en cuenta que ha de cerrar cada `\left` con el correspondiente `\right`, y que el tamaño se determina correctamente sólo si ambos se componen en la misma línea. Si no quiere que aparezca nada a la derecha, use ‘`\right.`’

$$1 + \left( \frac{1}{1-x^2} \right)^3$$

$$1 + \left( \frac{1}{1 - x^2} \right)^3$$

En algunos casos es necesario indicar el tamaño correcto de un delimitador matemático a mano, lo que puede hacerse con las órdenes `\big`, `\Big`, `\bigg` y `\Bigg` como prefijos de la mayoría de las órdenes de delimitador.<sup>4</sup>

$$\frac{(x+1)(x-1)}{(x+1)(x-1)} = 1$$

$$\left( (x+1)(x-1) \right)^2$$

Hay varias órdenes para introducir **tres puntos** en una fórmula. `\ldots` compone los puntos en la línea de base y `\cdots` los coloca centrados.

<sup>4</sup>Estas órdenes no funcionan bien si se usa una orden de cambio de tamaño, o si se indican las opciones `11pt` o `12pt`. Use los paquetes `exscale` o `amsmath` para corregir este comportamiento.



```
\begin{displaymath}
x_{\{1\}}, \ldots, x_{\{n\}} \quad \text{\quad} \\
x_{\{1\}} + \cdots + x_{\{n\}}
\end{displaymath}
```

$$x_1, \dots, x_n \qquad x_1 + \dots + x_n$$

Si los espacios en las fórmulas elegidos por T<sub>E</sub>X no son satisfactorios, pueden ajustarse insertando órdenes de espaciado especiales. Hay varias órdenes para espacios pequeños: `\`, para  $\frac{3}{18}$  de cuadratín (`\U`), `\:` para  $\frac{4}{18}$  de cuadratín (`\U`) y `\;` para  $\frac{5}{18}$  de cuadratín (`\U`). Es carácter espacio escapado `\_` genera un espacio de tamaño medio y `\quad` (`\_`) y `\qquad` (`\_`) producen espacios anchos. El tamaño de un cuadratín `\quad` corresponde a la anchura del carácter ‘M’ de la fundición actual. La orden `\!` produce un espacio negativo de  $-\frac{3}{18}$  de cuadratín (`\U`).

```
\newcommand{\ud}{\mathrm{d}}
\begin{displaymath}
\int\limits_{\int_0^1} g(x,y)
\quad , \quad \ud x, \quad \ud y
\end{displaymath}
en lugar de
\begin{displaymath}
\int\limits_{\int_0^1} g(x,y) \ud x \ud y
\end{displaymath}
```

$$\iint_D g(x,y) \, dx \, dy$$

en lugar de

$$\int \int_D g(x,y) \, dx \, dy$$

$\mathcal{A}\mathcal{M}\mathcal{S}\text{-}\mathcal{L}\mathcal{A}\mathcal{T}\mathcal{E}\mathcal{X}$  proporciona otra manera de afinar el espaciado entre múltiples signos integrales, mediante las órdenes `\iint`, `\iiint`, `\iiiiiint` y `\idotsint`. Con el paquete `amsmath` cargado, el ejemplo de arriba puede componerse así:

```
\newcommand{\ud}{\mathrm{d}}
\begin{displaymath}
\iint_{D} \, \, \ud x \, \, \ud y
\end{displaymath}
```

$$\iint_D \mathrm{d}x \mathrm{d}y$$

Vea el documento electrónico `testmath.tex` (distribuido con  $\mathcal{A}\mathcal{M}\mathcal{S}$ - $\text{\LaTeX}$ ) o el capítulo 8 de *The  $\text{\LaTeX}$  Companion* [3] para más detalles.

### 3.5. Material alineado verticalmente

Para componer **matrices**, use el entorno **array**. Funciona más o menos como el entorno **tabular**. La orden `\\` se usa para cambiar de fila.

```
\begin{displaymath}
\mathbf{X} =
\left( \begin{array}{ccc}
x_{11} & x_{12} & \ldots \\
x_{21} & x_{22} & \ldots \\
\vdots & \vdots & \ddots
\end{array} \right)
\end{displaymath}
```

$$\mathbf{X} = \begin{pmatrix} x_{11} & x_{12} & \dots \\ x_{21} & x_{22} & \dots \\ \vdots & \vdots & \ddots \end{pmatrix}$$

El entorno **array** también puede usarse para componer expresiones que tienen un delimitador grande usando “.” como un delimitador derecho (`\right`):

```
\begin{displaymath}
y = \left\{ \begin{array}{ll}
a & \text{si } d > c \\
b+x & \text{por la mañana} \\
l & \text{el resto del día}
\end{array} \right.
\end{displaymath}
```

$$y = \begin{cases} a & \text{si } d > c \\ b+x & \text{por la mañana} \\ l & \text{el resto del día} \end{cases}$$

Al igual que con el entorno **tabular**, puede también dibujar líneas en el entorno **array**, p.ej. separando los elementos de una matriz:

```
\begin{displaymath}
\left( \begin{array}{c|c}
1 & 2 \\ \hline
3 & 4
\end{array} \right)
\end{displaymath}
```

$$\left( \begin{array}{c|c} 1 & 2 \\ \hline 3 & 4 \end{array} \right)$$

Para fórmulas que ocupan varios renglones o para sistemas de ecuaciones, puede usar los entornos **eqnarray** y **eqnarray\*** en lugar de **equation**. En **eqnarray** cada renglón lleva un número de ecuación; en **eqnarray\*** no se numera ninguno.

Los entornos **eqnarray** y **eqnarray\*** funcionan como una tabla de tres columnas de la forma `{rcl}`, donde la columna del medio puede usarse para el signo *igual*, el signo *distinto* o cualquier otro signo que quiera poner. La orden `\\` cambia de renglón.

```
\begin{eqnarray}
f(x) &= & \cos x & \\
f'(x) &= & -\sin x & \\
\int_0^x f(y)dy &= & \sin x & \\
\end{eqnarray}
```

$$f(x) = \cos x \quad (3.5)$$

$$f'(x) = -\sin x \quad (3.6)$$

$$\int_0^x f(y)dy = \sin x \quad (3.7)$$

Tenga en cuenta que el espacio en ambos lados del signo *igual* es bastante grande. Puede reducirse poniendo `\setlength\arraycolsep{2pt}`, como en el siguiente ejemplo.

Las **ecuaciones largas** no se dividen automáticamente en trozos adecuados. El autor ha de indicar dónde partirlas y cuánto sangrar los trozos. Los siguientes dos métodos son los más habituales para conseguirlo.

```
{\setlength\arraycolsep{2pt}}
\begin{eqnarray}
\sin x &= & x - \frac{x^3}{3!} + \frac{x^5}{5!} - \\
&& \frac{x^7}{7!} + \cdots
\end{eqnarray}
```

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots \quad (3.8)$$

```
\begin{eqnarray}
\lefteqn{\cos x = 1} \\
& - \frac{x^2}{2!} + \\
& \frac{x^4}{4!} - \\
& \frac{x^6}{6!} + \cdots
\end{eqnarray}
```

$$\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \cdots \quad (3.9)$$

La orden `\nonumber` dice a  $\text{\LaTeX}$  que no genere un número para la correspondiente ecuación.

Puede resultar difícil conseguir ecuaciones alineadas en vertical de forma satisfactoria con estos métodos; el paquete `amsmath` proporciona un conjunto de alternativas más potentes. (Véanse los entornos `align`, `flalign`, `gather`, `multline` y `split`.)

### 3.6. Fantasmas

No podemos ver a los fantasmas, pero ocupan algo de espacio (al menos en la mente de mucha gente).  $\text{\LaTeX}$  no es diferente. Podemos aprovechar esto para conseguir ciertos efectos interesantes con el espaciado.

Al alinear verticalmente texto usando `^` y `_`  $\text{\LaTeX}$  a veces se pasa un poco de listo. Mediante la orden `\phantom` puede reservar espacio para caracteres

que no se muestran en la salida final. La forma más fácil de entenderlo es fijarse en los siguientes ejemplos.

```
\begin{displaymath}
{}^{\phantom{12}}_{12}\text{C}
\quad \text{frente a} \quad
{}^{\phantom{12}}_6\text{C}
\end{displaymath}
```

$${}^{12}_{6}\text{C} \quad \text{frente a} \quad {}^{12}_6\text{C}$$

```
\begin{displaymath}
\Gamma_{ij}^{\phantom{k}}
\quad \text{frente a} \quad
\Gamma_{ij}^k
\end{displaymath}
```

$$\Gamma_{ij}^{\phantom{k}} \quad \text{frente a} \quad \Gamma_{ij}^k$$

### 3.7. Tamaño de fundición en matemáticas

En modo matemático, TeX elige el tamaño de fundición según el contexto. Superíndices, por ejemplo, se componen con una fundición más pequeña. Si quiere componer parte de una ecuación con letra recta, no use la orden `\textrm`, porque el mecanismo de cambio de tamaño de fundición no funcionará, pues `\textrm` se escapa temporalmente a modo texto. Use `\mathrm` en su lugar para mantener activo el mecanismo de cambio. Pero esté atento, `\mathrm` sólo funcionará bien sobre argumentos cortos. Los espacios no estarán activos y los caracteres acentuados no funcionarán.<sup>5</sup>

```
\begin{equation}
2^{\text{nd}} \quad \quad \quad
2^{\mathrm{nd}} \quad \quad \quad
(3.10)
\end{equation}
```

$$2^{\text{nd}} \quad 2^{\mathrm{nd}} \quad (3.10)$$

A veces tendrá que indicar a L<sup>A</sup>T<sub>E</sub>X el tamaño de fundición correcto. En modo matemático, éste se establece con las siguientes cuatro órdenes:

```
\displaystyle (123), \textstyle (123), \scriptstyle (123) and
\scriptscriptstyle (123).
```

El cambio de estilo afecta también al modo en que se muestran los límites.

---

<sup>5</sup>El paquete `\mathcal{AMS-LATEX}` (`amsmath`) permite que la orden `\textrm` funcione con el cambio de tamaño.

```

\begin{displaymath}
\frac{\displaystyle
\sum_{i=1}^n(x_i-\overline{x})
(y_i-\overline{y})}
{\displaystyle\biggl[
\sum_{i=1}^n(x_i-\overline{x})^2
\sum_{i=1}^n(y_i-\overline{y})^2
\biggr]^{1/2}}
\end{displaymath}

```

$$\frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\left[ \sum_{i=1}^n (x_i - \bar{x})^2 \sum_{i=1}^n (y_i - \bar{y})^2 \right]^{1/2}}$$

Este es un ejemplo con corchetes más grandes que los que proporciona `\left[ \right]`. Las órdenes `\biggl` y `\biggr` se usan para paréntesis izquierdos y derechos respectivamente.

### 3.8. Lemas, teoremas, corolarios, ...

Al escribir documentos matemáticos, probablemente necesite una manera de componer “Lemas”, “Definiciones”, “Axiomas” y estructuras similares. Esto se hace con la orden `\newtheorem`.

```
\newtheorem{nombre}[contador]{texto}[sección]
```

El argumento *nombre* es una palabra corta usada para identificar el tipo de “teorema”. Con el argumento *texto* se define el nombre real del “teorema”, que aparecerá en el documento final.

Los argumentos entre corchetes son opcionales. Se usan ambos para indicar la numeración usada en el “teorema”. Use el argumento *contador* para indicar el *nombre* de un “teorema” declarado con anterioridad. El nuevo “teorema” se numerará en la misma secuencia. El argumento *sección* le permite indicar una unidad de sección de la cual el “teorema” tomará sus números.

Tras ejecutar la orden `\newtheorem` en el preámbulo de su documento, puede usar la siguiente orden dentro del documento.

```

\begin{nombre}[texto]
Este es mi interesante teorema
\end{nombre}

```

El paquete `amsthm` proporciona la orden `\newtheoremstyle{estilo}` que le permite definir sobre qué va el teorema escogiendo entre tres estilos predefinidos: **definition** (título en negrita, cuerpo en recta), **plain** (título en negrita, cuerpo en cursiva) o **remark** (título en cursiva, cuerpo en recta).

Esto debería bastar como teoría. Los siguientes ejemplos deberían despejar las dudas restantes, y dejar claro que el entorno `\newtheorem` es demasiado complejo de entender.

Primero defina los teoremas:

```
\theoremstyle{definition} \newtheorem{ley}{Ley}
\theoremstyle{plain}      \newtheorem{jurado}[ley]{Jurado}
\theoremstyle{remark}     \newtheorem*{marg}{Margarita}
```

```
\begin{ley} \label{ley:caja}
No esconder en la caja negra
\end{ley}
\begin{jurado}[Los Doce]
¡Podría ser usted! Cuidado y
vea la ley~\ref{ley:caja}
\end{jurado}
\begin{marg}No, No, No\end{marg}
```

**Ley 1.** No esconder en la caja negra

**Jurado 2** (Los Doce). *¡Podría ser usted!*  
*Cuidado y vea la ley [1](#)*

*Margarita.* No, No, No

El teorema “Jurado” usa el mismo contador que el teorema “Ley”, así que le corresponde un número en secuencia con las otras “Leyes”. El argumento entre corchetes se usa para indicar un título o algo similar para el teorema.

```
\flushleft
\newtheorem{mur}{Murphy}[section]
\begin{mur}
Si hay dos o más formas de
hacer algo, y una de ellas
puede resultar catastrófica,
entonces alguien la escogerá.
\end{mur}
```

*Murphy 3.8.1.* Si hay dos o más formas de hacer algo, y una de ellas puede resultar catastrófica, entonces alguien la escogerá.

El teorema “Murphy” recibe un número que está ligado al número de la sección actual. Podría usar otra unidad, como por ejemplo `chapter` o `subsection`.

El paquete `amsthm` también proporciona `proof` para demostraciones.

```
\begin{proof}
Trivial, use
\mathrel{E=mc^2}
\end{proof}
```

*Demostración.* Trivial, use

$$E = mc^2$$

□

Con la orden `\qedhere` puede mover el ‘símbolo de fin de demostración’ para las situaciones en que terminaría solo en un renglón.

```
\begin{proof}
Trivial, use
\mathrel{E=mc^2} \qedhere
\end{proof}
```

*Demostración.* Trivial, use

$$E = mc^2$$

□

### 3.9. Símbolos en negrita

Es bastante difícil conseguir símbolos en negrita en  $\text{\LaTeX}$ ; probablemente es a propósito, pues los compositores aficionados tienden a abusar de ellos. La orden de cambio de fundición `\mathbf` da letras en negrita, pero éstas son rectas mientras que los símbolos matemáticos son normalmente en cursiva. Hay una orden `\boldmath`, pero *sólo puede usarse fuera del modo matemático*. Funciona también para símbolos.

```
\begin{displaymath}
\mu, M \quad \mathbf{M} \quad \mu, M
\mbox{\boldmath $\mu, M$}
\end{displaymath}
```



Fíjese en que la coma también es negrita, lo que puede no ser lo que se pretende.

El paquete `amssbsy` (incluido por `amsmath`) y también el `bm` facilitan la labor al proporcionar la orden `\boldsymbol`.

```
\begin{displaymath}
\mu, M \quad \boldsymbol{\mu}, \boldsymbol{M}
\end{displaymath}
```



### 3.10. Lista de símbolos matemáticos

Los siguientes cuadros muestran todos los símbolos normalmente accesibles desde *modo matemático*.

Para usar los símbolos listados en los cuadros 3.11-3.15,<sup>6</sup> debe cargarse el paquete `amssymb` en el preámbulo del documento y las fundiciones de la AMS deben estar instaladas en el sistema. Si el paquete y las fundiciones AMS no están instaladas en su sistema, mire en `macros/latex/required/amslatex`. Una lista de símbolos aun más completa se puede hallar en `info/symbols/comprehensive`.

Cuadro 3.1: Acentos en modo matemático.

$\hat{a}$	<code>\hat{a}</code>	$\check{a}$	<code>\check{a}</code>	$\tilde{a}$	<code>\tilde{a}</code>
$\grave{a}$	<code>\grave{a}</code>	$\dot{a}$	<code>\dot{a}</code>	$\ddot{a}$	<code>\ddot{a}</code>
$\bar{a}$	<code>\bar{a}</code>	$\vec{a}$	<code>\vec{a}</code>	$\widehat{A}$	<code>\widehat{A}</code>
$\acute{a}$	<code>\acute{a}</code>	$\breve{a}$	<code>\breve{a}</code>	$\widetilde{A}$	<code>\widetilde{A}</code>

Cuadro 3.2: Letras griegas.

$\alpha$	<code>\alpha</code>	$\theta$	<code>\theta</code>	$o$	<code>o</code>	$\upsilon$	<code>\upsilon</code>
$\beta$	<code>\beta</code>	$\vartheta$	<code>\vartheta</code>	$\pi$	<code>\pi</code>	$\phi$	<code>\phi</code>
$\gamma$	<code>\gamma</code>	$\iota$	<code>\iota</code>	$\varpi$	<code>\varpi</code>	$\varphi$	<code>\varphi</code>
$\delta$	<code>\delta</code>	$\kappa$	<code>\kappa</code>	$\rho$	<code>\rho</code>	$\chi$	<code>\chi</code>
$\epsilon$	<code>\epsilon</code>	$\lambda$	<code>\lambda</code>	$\varrho$	<code>\varrho</code>	$\psi$	<code>\psi</code>
$\varepsilon$	<code>\varepsilon</code>	$\mu$	<code>\mu</code>	$\sigma$	<code>\sigma</code>	$\omega$	<code>\omega</code>
$\zeta$	<code>\zeta</code>	$\nu$	<code>\nu</code>	$\varsigma$	<code>\varsigma</code>		
$\eta$	<code>\eta</code>	$\xi$	<code>\xi</code>	$\tau$	<code>\tau</code>		
$\Gamma$	<code>\Gamma</code>	$\Lambda$	<code>\Lambda</code>	$\Sigma$	<code>\Sigma</code>	$\Psi$	<code>\Psi</code>
$\Delta$	<code>\Delta</code>	$\Xi$	<code>\Xi</code>	$\Upsilon$	<code>\Upsilon</code>	$\Omega$	<code>\Omega</code>
$\Theta$	<code>\Theta</code>	$\Pi$	<code>\Pi</code>	$\Phi$	<code>\Phi</code>		

<sup>6</sup>Estos cuadros provienen de `symbols.tex` de David Carlisle y fueron cambiando mucho según las sugerencias de Josef Tkadlec.



Cuadro 3.3: Relaciones binarias.

Puede negar los símbolos siguientes prefijándolos con la orden `\not`.

$<$	<code>&lt;</code>	$>$	<code>&gt;</code>	$=$	<code>=</code>
$\leq$	<code>\leq</code> or <code>\le</code>	$\geq$	<code>\geq</code> or <code>\ge</code>	$\equiv$	<code>\equiv</code>
$\ll$	<code>\ll</code>	$\gg$	<code>\gg</code>	$\dot{=}$	<code>\doteq</code>
$\prec$	<code>\prec</code>	$\succ$	<code>\succ</code>	$\sim$	<code>\sim</code>
$\preceq$	<code>\preceq</code>	$\succeq$	<code>\succeq</code>	$\simeq$	<code>\simeq</code>
$\subset$	<code>\subset</code>	$\supset$	<code>\supset</code>	$\approx$	<code>\approx</code>
$\subseteq$	<code>\subseteq</code>	$\supseteq$	<code>\supseteq</code>	$\cong$	<code>\cong</code>
$\sqsubset$	<code>\sqsubset</code> <sup>a</sup>	$\sqsupset$	<code>\sqsupset</code> <sup>a</sup>	$\Join$	<code>\Join</code> <sup>a</sup>
$\sqsubseteq$	<code>\sqsubseteq</code>	$\sqsupseteq$	<code>\sqsupseteq</code>	$\bowtie$	<code>\bowtie</code>
$\in$	<code>\in</code>	$\ni$	<code>\ni</code> , <code>\owns</code>	$\propto$	<code>\propto</code>
$\vdash$	<code>\vdash</code>	$\dashv$	<code>\dashv</code>	$\models$	<code>\models</code>
$\mid$	<code>\mid</code>	$\parallel$	<code>\parallel</code>	$\perp$	<code>\perp</code>
$\smile$	<code>\smile</code>	$\frown$	<code>\frown</code>	$\asymp$	<code>\asymp</code>
$:$	<code>:</code>	$\notin$	<code>\notin</code>	$\neq$	<code>\neq</code> or <code>\ne</code>

<sup>a</sup> Use el paquete `latexsym` para acceder a este símbolo

Cuadro 3.4: Operadores binarios.

$+$	<code>+</code>	$-$	<code>-</code>	
$\pm$	<code>\pm</code>	$\mp$	<code>\mp</code>	$\triangleleft$ <code>\triangleleft</code>
$\cdot$	<code>\cdot</code>	$\div$	<code>\div</code>	$\triangleright$ <code>\triangleright</code>
$\times$	<code>\times</code>	$\setminus$	<code>\setminus</code>	$\star$ <code>\star</code>
$\cup$	<code>\cup</code>	$\cap$	<code>\cap</code>	$\ast$ <code>\ast</code>
$\sqcup$	<code>\sqcup</code>	$\sqcap$	<code>\sqcap</code>	$\circ$ <code>\circ</code>
$\vee$ , <code>\lor</code>	<code>\vee</code> , <code>\lor</code>	$\wedge$ <code>\wedge</code> , <code>\land</code>	<code>\wedge</code> , <code>\land</code>	$\bullet$ <code>\bullet</code>
$\oplus$	<code>\oplus</code>	$\ominus$	<code>\ominus</code>	$\diamond$ <code>\diamond</code>
$\odot$	<code>\odot</code>	$\oslash$	<code>\oslash</code>	$\uplus$ <code>\uplus</code>
$\otimes$	<code>\otimes</code>	$\bigcirc$	<code>\bigcirc</code>	$\amalg$ <code>\amalg</code>
$\triangleup$	<code>\triangleup</code>	$\triangledown$	<code>\triangledown</code>	$\dagger$ <code>\dagger</code>
$\lhd$ <sup><i>a</i></sup>	<code>\lhd</code> <sup><i>a</i></sup>	$\rhd$ <sup><i>a</i></sup>	<code>\rhd</code> <sup><i>a</i></sup>	$\ddagger$ <code>\ddagger</code>
$\unlhd$ <sup><i>a</i></sup>	<code>\unlhd</code> <sup><i>a</i></sup>	$\unrhd$ <sup><i>a</i></sup>	<code>\unrhd</code> <sup><i>a</i></sup>	$\wr$ <code>\wr</code>

Cuadro 3.5: Operadores GRANDES.

$\sum$	<code>\sum</code>	$\bigcup$	<code>\bigcup</code>	$\bigvee$	<code>\bigvee</code>
$\prod$	<code>\prod</code>	$\bigcap$	<code>\bigcap</code>	$\bigwedge$	<code>\bigwedge</code>
$\coprod$	<code>\coprod</code>	$\bigsqcup$	<code>\bigsqcup</code>	$\biguplus$	<code>\biguplus</code>
$\int$	<code>\int</code>	$\oint$	<code>\oint</code>	$\bigodot$	<code>\bigodot</code>
$\oplus$	<code>\bigoplus</code>	$\otimes$	<code>\bigotimes</code>		

Cuadro 3.6: Flechas.

$\leftarrow$	<code>\leftarrow</code> o <code>\gets</code>	$\longleftarrow$	<code>\longleftarrow</code>
$\rightarrow$	<code>\rightarrow</code> o <code>\to</code>	$\longrightarrow$	<code>\longrightarrow</code>
$\leftrightarrow$	<code>\leftrightarrow</code>	$\longleftrightarrow$	<code>\longleftrightarrow</code>
$\Leftarrow$	<code>\Leftarrow</code>	$\Longleftarrow$	<code>\Longleftarrow</code>
$\Rightarrow$	<code>\Rightarrow</code>	$\Longrightarrow$	<code>\Longrightarrow</code>
$\Leftrightarrow$	<code>\Leftrightarrow</code>	$\Longleftrightarrow$	<code>\Longleftrightarrow</code>
$\mapsto$	<code>\mapsto</code>	$\longmapsto$	<code>\longmapsto</code>
$\hookleftarrow$	<code>\hookleftarrow</code>	$\hookrightarrow$	<code>\hookrightarrow</code>
$\leftharpoonup$	<code>\leftharpoonup</code>	$\rightharpoonup$	<code>\rightharpoonup</code>
$\leftharpoondown$	<code>\leftharpoondown</code>	$\rightharpoondown$	<code>\rightharpoondown</code>
$\rightleftharpoons$	<code>\rightleftharpoons</code>	$\iff$ (espacios mayores)	<code>\iff</code> (espacios mayores)
$\uparrow$	<code>\uparrow</code>	$\downarrow$	<code>\downarrow</code>
$\updownarrow$	<code>\updownarrow</code>	$\Uparrow$	<code>\Uparrow</code>
$\Downarrow$	<code>\Downarrow</code>	$\Updownarrow$	<code>\Updownarrow</code>
$\nearrow$	<code>\nearrow</code>	$\searrow$	<code>\searrow</code>
$\swarrow$	<code>\swarrow</code>	$\nwarrow$	<code>\nwarrow</code>
$\leadsto$	<code>\leadsto</code> <sup>a</sup>		

<sup>a</sup> Use el paquete `latexsym` para acceder a este símbolo

Cuadro 3.7: Delimitadores.

$($	$($	$)$	$)$	$\uparrow$	<code>\uparrow</code>
$[$	<code>[</code> o <code>\lbrack</code>	$]$	<code>] o \rbrack</code>	$\downarrow$	<code>\downarrow</code>
$\{$	<code>\{</code> o <code>\lbrace</code>	$\}$	<code>\} o \rbrace</code>	$\updownarrow$	<code>\updownarrow</code>
$\langle$	<code>\langle</code>	$\rangle$	<code>\rangle</code>	$ $	<code>  o \vert</code>
$\lfloor$	<code>\lfloor</code>	$\rfloor$	<code>\rfloor</code>	$\lceil$	<code>\lceil</code>
$/$	<code>/</code>	$\backslash$	<code>\backslash</code>	$\Updownarrow$	<code>\Updownarrow</code>
$\Uparrow$	<code>\Uparrow</code>	$\Downarrow$	<code>\Downarrow</code>	$\ $	<code>\  o \Vert</code>
$\rceil$	<code>\rceil</code>				

Cuadro 3.8: Delimitadores grandes.

$\left($	<code>\lgroup</code>	$\right)$	<code>\rgroup</code>	$\int$	<code>\lmoustache</code>
$\downarrow$	<code>\arrowvert</code>	$\Downarrow$	<code>\Arrowvert</code>	$\} $	<code>\bracevert</code>
$\left\{$	<code>\rmoustache</code>				

Cuadro 3.9: Símbolos variados.

$\dots$	<code>\dots</code>	$\cdots$	<code>\cdots</code>	$\vdots$	<code>\vdots</code>	$\ddots$	<code>\ddots</code>
$\hbar$	<code>\hbar</code>	$\imath$	<code>\imath</code>	$\jmath$	<code>\jmath</code>	$\ell$	<code>\ell</code>
$\Re$	<code>\Re</code>	$\Im$	<code>\Im</code>	$\aleph$	<code>\aleph</code>	$\wp$	<code>\wp</code>
$\forall$	<code>\forall</code>	$\exists$	<code>\exists</code>	$\mho$	<code>\mho</code>	$\partial$	<code>\partial</code>
$'$	<code>'</code>	$'$	<code>\prime</code>	$\emptyset$	<code>\emptyset</code>	$\infty$	<code>\infty</code>
$\nabla$	<code>\nabla</code>	$\triangle$	<code>\triangle</code>	$\Box$	<code>\Box</code>	$\diamond$	<code>\Diamond</code>
$\bot$	<code>\bot</code>	$\top$	<code>\top</code>	$\angle$	<code>\angle</code>	$\surd$	<code>\surd</code>
$\diamondsuit$	<code>\diamondsuit</code>	$\heartsuit$	<code>\heartsuit</code>	$\clubsuit$	<code>\clubsuit</code>	$\spadesuit$	<code>\spadesuit</code>
$\neg$	<code>\neg</code> or <code>\lnot</code>	$\flat$	<code>\flat</code>	$\natural$	<code>\natural</code>	$\sharp$	<code>\sharp</code>

<sup>a</sup> Use el paquete latexsym para acceder a este símbolo

Cuadro 3.10: Símbolos no matemáticos.

Estos símbolos pueden usarse también en modo texto.

$\dagger$	<code>\dag</code>	$\S$	<code>\S</code>	$\copyright$	<code>\copyright</code>	$\textregistered$	<code>\textregistered</code>
$\ddagger$	<code>\ddag</code>	$\P$	<code>\P</code>	$\pounds$	<code>\pounds</code>	$\%$	<code>\%</code>

Cuadro 3.11: Delimitadores AMS.

$\ulcorner$	<code>\ulcorner</code>	$\urcorner$	<code>\urcorner</code>	$\llcorner$	<code>\llcorner</code>	$\lrcorner$	<code>\lrcorner</code>
$\lvert$	<code>\lvert</code>	$\rvert$	<code>\rvert</code>	$\lVert$	<code>\lVert</code>	$\rVert$	<code>\rVert</code>

Cuadro 3.12: Símbolos AMS griegos y hebreos.

$\digamma$	<code>\digamma</code>	$\varkappa$	<code>\varkappa</code>	$\beth$	<code>\beth</code>	$\gimel$	<code>\gimel</code>	$\daleth$	<code>\daleth</code>
------------	-----------------------	-------------	------------------------	---------	--------------------	----------	---------------------	-----------	----------------------

Cuadro 3.13: Relaciones binarias AMS.

$\lessdot$	<code>\lessdot</code>	$\gtrdot$	<code>\gtrdot</code>	$\doteqdot$	<code>\doteqdot</code>
$\leqslant$	<code>\leqslant</code>	$\geqslant$	<code>\geqslant</code>	$\risingdotseq$	<code>\risingdotseq</code>
$\eqslantless$	<code>\eqslantless</code>	$\eqslantgtr$	<code>\eqslantgtr</code>	$\fallingdotseq$	<code>\fallingdotseq</code>
$\leqq$	<code>\leqq</code>	$\geqq$	<code>\geqq</code>	$\eqcirc$	<code>\eqcirc</code>
$\lll$ o $\llless$	<code>\lll</code> o <code>\llless</code>	$\ggg$	<code>\ggg</code>	$\circeq$	<code>\circeq</code>
$\lesssim$	<code>\lesssim</code>	$\gtrsim$	<code>\gtrsim</code>	$\triangleq$	<code>\triangleq</code>
$\lessapprox$	<code>\lessapprox</code>	$\gtrapprox$	<code>\gtrapprox</code>	$\bumpeq$	<code>\bumpeq</code>
$\lessgtr$	<code>\lessgtr</code>	$\gtrless$	<code>\gtrless</code>	$\Bumpeq$	<code>\Bumpeq</code>
$\lesseqgtr$	<code>\lesseqgtr</code>	$\gtreqless$	<code>\gtreqless</code>	$\thicksim$	<code>\thicksim</code>
$\lesseqqgtr$	<code>\lesseqqgtr</code>	$\gtreqqless$	<code>\gtreqqless</code>	$\thickapprox$	<code>\thickapprox</code>
$\preccurlyeq$	<code>\preccurlyeq</code>	$\succcurlyeq$	<code>\succcurlyeq</code>	$\approxeq$	<code>\approxeq</code>
$\curlyeqprec$	<code>\curlyeqprec</code>	$\curlyeqsucc$	<code>\curlyeqsucc</code>	$\backsim$	<code>\backsim</code>
$\precsim$	<code>\precsim</code>	$\succsim$	<code>\succsim</code>	$\backsimeq$	<code>\backsimeq</code>
$\precapprox$	<code>\precapprox</code>	$\succapprox$	<code>\succapprox</code>	$\vDash$	<code>\vDash</code>
$\subseteq$	<code>\subseteq</code>	$\supseteq$	<code>\supseteq</code>	$\Vdash$	<code>\Vdash</code>
$\shortparallel$	<code>\shortparallel</code>	$\Supset$	<code>\Supset</code>	$\Vvdash$	<code>\Vvdash</code>
$\blacktriangleleft$	<code>\blacktriangleleft</code>	$\sqsupset$	<code>\sqsupset</code>	$\backepsilon$	<code>\backepsilon</code>
$\vartriangleright$	<code>\vartriangleright</code>	$\because$	<code>\because</code>	$\varpropto$	<code>\varpropto</code>
$\blacktriangleright$	<code>\blacktriangleright</code>	$\Subset$	<code>\Subset</code>	$\between$	<code>\between</code>
$\trianglerighteq$	<code>\trianglerighteq</code>	$\smallfrown$	<code>\smallfrown</code>	$\pitchfork$	<code>\pitchfork</code>
$\vartriangleleft$	<code>\vartriangleleft</code>	$\shortmid$	<code>\shortmid</code>	$\smallsmile$	<code>\smallsmile</code>
$\trianglelefteq$	<code>\trianglelefteq</code>	$\therefore$	<code>\therefore</code>	$\sqsubset$	<code>\sqsubset</code>

Cuadro 3.14: Flechas AMS.

$\dashleftarrow$	<code>\dashleftarrow</code>	$\dashrightarrow$	<code>\dashrightarrow</code>
$\Leftrightarrow$	<code>\leftleftarrows</code>	$\Rrightarrow$	<code>\rightrightarrows</code>
$\Leftrightarrow$	<code>\leftrightharpoons</code>	$\Rrightarrow$	<code>\rightleftarrows</code>
$\Leftrightarrow$	<code>\Lleftarrow</code>	$\Rrightarrow$	<code>\Rrightarrow</code>
$\Leftrightarrow$	<code>\twoheadleftarrow</code>	$\Rrightarrow$	<code>\twoheadrightarrow</code>
$\Leftrightarrow$	<code>\leftarrowtail</code>	$\Rrightarrow$	<code>\rightarrowtail</code>
$\Leftrightarrow$	<code>\leftrightharpoons</code>	$\Rrightarrow$	<code>\rightleftharpoons</code>
$\Leftrightarrow$	<code>\Lsh</code>	$\Rrightarrow$	<code>\Rsh</code>
$\Leftrightarrow$	<code>\looparrowleft</code>	$\Rrightarrow$	<code>\looparrowright</code>
$\Leftrightarrow$	<code>\curvearrowleft</code>	$\Rrightarrow$	<code>\curvearrowright</code>
$\Leftrightarrow$	<code>\circlearrowleft</code>	$\Rrightarrow$	<code>\circlearrowright</code>
$\Leftrightarrow$	<code>\multimap</code>	$\Rrightarrow$	<code>\upuparrows</code>
$\Leftrightarrow$	<code>\downdownarrows</code>	$\Rrightarrow$	<code>\upharpoonleft</code>
$\Leftrightarrow$	<code>\upharpoonright</code>	$\Rrightarrow$	<code>\downharpoonright</code>
$\Leftrightarrow$	<code>\rightsquigarrow</code>	$\Rrightarrow$	<code>\leftrightsquigarrow</code>

Cuadro 3.15: Relaciones binarias y flechas negadas AMS.

$\nless$	$\ngtr$	$\varsubsetneqq$
$\lneq$	$\gneq$	$\varsupsetneqq$
$\nleq$	$\ngeq$	$\nsubseteqeq$
$\nleqslant$	$\ngeqslant$	$\nsupseteqeq$
$\lneqq$	$\gneqq$	$\nmid$
$\lvertneqq$	$\gvertneqq$	$\nparallel$
$\nleqq$	$\ngeqq$	$\nshortmid$
$\lnsim$	$\gnsim$	$\nshortparallel$
$\lnapprox$	$\gnapprox$	$\nsim$
$\nprec$	$\nsucc$	$\ncong$
$\npreceq$	$\nsucceq$	$\nvdash$
$\precneqq$	$\succneqq$	$\nvDash$
$\precnsim$	$\succnsim$	$\nVdash$
$\precnapprox$	$\succnapprox$	$\nVDash$
$\subsetneq$	$\supsetneq$	$\ntriangleleft$
$\varsubsetneq$	$\varsupsetneq$	$\ntriangleright$
$\nsubseteq$	$\nsupseteq$	$\ntrianglelefteq$
$\subseteqeq$	$\supseteqeq$	$\ntrianglerighteq$
$\nleftarrow$	$\rightarrow$	$\nleftrightarrow$
$\nLeftarrow$	$\Rightarrow$	$\nLeftrightarrow$

Cuadro 3.16: Operadores binarios AMS.

$\dot{+}$	$\dot{+}$	$\dot{+}$
$\ltimes$	$\rtimes$	$\divideontimes$
$\doublecup$	$\doublecap$	$\smallsetminus$
$\veebar$	$\bar{\wedge}$	$\doublebarwedge$
$\boxplus$	$\boxminus$	$\circleddash$
$\boxtimes$	$\boxdot$	$\circledcirc$
$\intercal$	$\circledast$	$\rightthreetimes$
$\curlyvee$	$\curlywedge$	$\leftthreetimes$

$\hbar$	<code>\hbar</code>	$\hslash$	<code>\hslash</code>	$\Bbbk$	<code>\Bbbk</code>
$\square$	<code>\square</code>	$\blacksquare$	<code>\blacksquare</code>	$\textcircled{S}$	<code>\circledS</code>
$\triangle$	<code>\vartriangle</code>	$\blacktriangle$	<code>\blacktriangle</code>	$\complement$	<code>\complement</code>
$\nabla$	<code>\triangledown</code>	$\blacktriangledown$	<code>\blacktriangledown</code>	$\Game$	<code>\Game</code>
$\lozenge$	<code>\lozenge</code>	$\blacklozenge$	<code>\blacklozenge</code>	$\star$	<code>\bigstar</code>
$\angle$	<code>\angle</code>	$\measuredangle$	<code>\measuredangle</code>		
$\diagup$	<code>\diagup</code>	$\diagdown$	<code>\diagdown</code>	$\backprime$	<code>\backprime</code>
$\nexists$	<code>\nexists</code>	$\Finv$	<code>\Finv</code>	$\varnothing$	<code>\varnothing</code>
$\eth$	<code>\eth</code>	$\sphericalangle$	<code>\sphericalangle</code>	$\mho$	<code>\mho</code>

Ejemplo	Orden	Paquete requerido
ABCDEabcde1234	<code>\mathrm{ABCDE abcde 1234}</code>	ninguno
<i>ABCDEabcde1234</i>	<code>\mathit{ABCDE abcde 1234}</code>	ninguno
<i>ABCDEF</i> abcde1234	<code>\mathnormal{ABCDE abcde 1234}</code>	ninguno
$\mathcal{A}\mathcal{B}\mathcal{C}\mathcal{D}\mathcal{E}$	<code>\mathcal{ABCDE abcde 1234}</code>	ninguno
$\mathscr{A}\mathscr{B}\mathscr{C}\mathscr{D}\mathscr{E}$	<code>\mathscr{ABCDE abcde 1234}</code>	mathrsfs
$\frac{A\mathcal{B}\mathcal{C}\mathcal{D}E}{abcde1234}$	<code>\mathfrak{ABCDE abcde 1234}</code>	amsfonts o amssymb
$\mathbb{A}\mathbb{B}\mathbb{C}\mathbb{D}\mathbb{E}\mathbb{F}\mathbb{G}\mathbb{H}\mathbb{I}\mathbb{J}\mathbb{K}\mathbb{L}\mathbb{M}\mathbb{N}$	<code>\mathbb{ABCDE abcde 1234}</code>	amsfonts o amssymb

## Capítulo 4

# Especialidades

Al juntar las partes de un documento grande,  $\text{\LaTeX}$  lo ayudará con algunas funciones especiales como generación de índices, gestión de la bibliografía, y otras cosas. Una descripción mucho más completa de las especialidades y mejoras posibles con  $\text{\LaTeX}$  puede hallarse en  *$\text{\LaTeX}$  Manual* [1] y *The  $\text{\LaTeX}$  Companion* [3].

### 4.1. Inclusión de Encapsulated PostScript

$\text{\LaTeX}$  proporciona las facilidades básicas para trabajar con objetos deslizantes, como imágenes o gráficos, mediante los entornos `figure` y `table`.

Hay varias maneras de generar los gráficos reales con  $\text{\LaTeX}$  básico o con un paquete de extensión de  $\text{\LaTeX}$ ; algunos de ellos se describen en el capítulo 5. Para más información sobre el asunto diríjase a *The  $\text{\LaTeX}$  Companion* [3] y a  *$\text{\LaTeX}$  Manual* [1].

Una manera mucha más fácil de incorporar gráficos en un documento es generarlos con un programa especializado<sup>1</sup> y después incluir los gráficos acabados en el documento. De nuevo, los paquetes de  $\text{\LaTeX}$  ofrecen muchas formas de hacerlo, pero esta introducción solamente comentará el uso de gráficos Encapsulated POSTSCRIPT (EPS), porque es bastante fácil de hacer y de uso muy habitual. Para usar dibujos en formato EPS, debe tener una impresora POSTSCRIPT<sup>2</sup> disponible.

Se proporciona un buen conjunto de órdenes para incluir gráficos en el paquete `graphicx` de D. P. Carlisle. Es parte de una familia entera de paquetes llamada “graphics”.<sup>3</sup>

Suponiendo que está trabajando en un sistema con una impresora POSTSCRIPT disponible y con el paquete `graphicx` instalado, puede usar la siguiente

---

<sup>1</sup>Como XFig, Scribus, OpenOffice Draw, Gnuplot, ...

<sup>2</sup>Otra posibilidad de ver un POSTSCRIPT es usar el programa GHOSTSCRIPT disponible en `support/ghostscript`. Los usuarios de ReactOS o Windows pueden buscar GSVIEW.

<sup>3</sup>`macros/latex/required/graphics`



guía paso a paso para incluir un dibujo en su documento:

1. Exporte el dibujo desde su programa de gráficos en formato EPS.<sup>4</sup>
2. Cargue el paquete `graphicx` en el preámbulo del fichero de entrada con

```
\usepackage[controlador]{graphicx}
```

donde *controlador* es el nombre de su programa conversor “dvi a post-script”. El programa más usado se llama `dvips`. Se requiere el nombre del controlador, porque no hay una norma para la inclusión de gráficos en  $\text{\TeX}$ . Sabido el nombre del *controlador*, el paquete `graphicx` puede escoger el método correcto para insertar la información sobre los gráficos en el fichero `.dvi`, de forma que la impresora la entienda y pueda incluir correctamente el fichero `.eps`.

3. Use la orden

```
\includegraphics[clave=valor, ...]{fichero}
```

para incluir el *fichero* en su documento. El parámetro opcional acepta una lista separada por comas de *claves* y *valores* asociados. Las *claves* pueden usarse para alterar la anchura, altura y giro del gráfico incluido. El cuadro 4.1 lista las claves más importantes.

Cuadro 4.1: Nombres de claves para el paquete `graphicx`.

<b>width</b>	escala el gráfico a la anchura indicada
<b>height</b>	escala el gráfico a la altura indicada
<b>angle</b>	gira el gráfico en sentido antihorario
<b>scale</b>	escala el gráfico

<sup>4</sup>Si su programa no puede exportar al formato EPS, puede intentar instalar un controlador de impresora POSTSCRIPT (como Apple LaserWriter, por ejemplo) y entonces imprimir a un fichero a través de ese controlador. Con suerte tal fichero tendrá formato EPS. Tenga en cuenta que un EPS no debe contener más de una página. Algunos controladores de impresora pueden configurarse explícitamente para producir formato EPS.

El siguiente código de ejemplo puede ayudar a aclarar las cosas:

```
\begin{figure}
\centering
\includegraphics[angle=90,
                  width=0.5\textwidth]{prueba}
\caption{Esto es una prueba.}
\end{figure}
```

Incluye el gráfico almacenado en el fichero `prueba.eps`. El gráfico *primero* se gira según un ángulo de 90 grados sexagesimales y *después* se escala a la anchura final de 05 veces la anchura de un párrafo normal. La altura final estará en proporción según las dimensiones originales, porque ninguna altura se indica explícitamente. Los parámetros de altura y anchura pueden indicarse como medidas absolutas. Mire en el cuadro 6.5 de la página 123 para más información. Si quiere saber más sobre este asunto, lea [9] y [13].

## 4.2. Bibliografía

Puede crear una bibliografía con el entorno `thebibliography`. Cada entrada empieza con

`\bibitem[etiqueta]{marcador}`

El *marcador* se usa para citar el libro o artículo desde el documento.

`\cite{marcador}`

Si no usa la opción *etiqueta*, las entradas se numerarán automáticamente. El parámetro tras la orden `\begin{thebibliography}` define cuánto espacio reservar para el número de las etiquetas. En el próximo ejemplo, `{99}` dice a L<sup>A</sup>T<sub>E</sub>X que espere que ninguno de esos números será más ancho que el número 99.

```
Partl~\cite{pa} ha  
propuesto que...  
\begin{thebibliography}{99}  
\bibitem{pa} H.~Partl:  
\emph{\TeX{} in German},  
TUGboat, Volumen~9, Núm.~1 (1988).  
\end{thebibliography}
```

Partl [1] ha propuesto que...

## Bibliografía

- [1] H. Partl: *T<sub>E</sub>X in German*, TUGboat,  
Volumen 9, Núm. 1 (1988).

Para proyectos mayores, podría convenirle el programa BibT<sub>E</sub>X. BibT<sub>E</sub>X se incluye en la mayoría de las distribuciones T<sub>E</sub>X. Le permite mantener una base de datos bibliográfica y después extraer las referencias relevantes a lo que cite en su artículo. La presentación visual de las bibliografías generadas con BibT<sub>E</sub>X se basa en un concepto de hojas de estilo que le permiten crear bibliografías que sigan un amplio rango de diseños establecidos.

Cuadro 4.2: Ejemplos de sintaxis de las claves para el índice.

Ejemplo	En el índice	Comentario
<code>\index{hola}</code>	hola, 1	Entrada básica
<code>\index{hola!Pedro}</code>	Pedro, 3	Subentrada bajo ‘hola’
<code>\index{Sam@\textsl{Sam}}</code>	<i>Sam</i> , 2	Entrada con formato
<code>\index{Lin@\textbf{Lin}}</code>	<b>Lin</b> , 7	Ídem
<code>\index{Yeni textbf}</code>	Yeni, <b>3</b>	Núm. pág. con formato
<code>\index{Pepe textit}</code>	Pepe, 5	Ídem
<code>\index{Jose@Jos\'e}</code>	José, 4	Uso de acentos

### 4.3. Índices

Una parte muy útil de muchos libros es su índice. Con  $\text{\LaTeX}$  y el programa de soporte `makeindex`,<sup>5</sup> se puede generar un índice fácilmente. Esta introducción le explicará sólo las órdenes básicas de generación de un índice. Para un visión más profunda, diríjase a *The  $\text{\LaTeX}$  Companion* [3].

Para habilitar la capacidad de indexado de  $\text{\LaTeX}$ , se debe cargar el paquete `makeidx` en el preámbulo con:

```
\usepackage{makeidx}
```

y las órdenes especiales de indexado deben habilitarse poniendo la orden

```
\makeindex
```

en el preámbulo del fichero de entrada.

El contenido del índice se indica con órdenes

```
\index{clave}
```

donde *clave* es la entrada del índice. Introduzca las órdenes en los puntos del texto adonde quiera que apunten las entradas del índice final. El cuadro 4.2 explica la sintaxis del argumento *clave* con varios ejemplos.

Cuando el fichero de entrada se procesa con  $\text{\LaTeX}$ , cada orden `\index` escribe una entrada apropiada del índice, junto con el número de página actual, a un fichero especial. El fichero tiene el mismo nombre que el fichero de entrada  $\text{\LaTeX}$ , pero una extensión diferente (`.idx`). Este fichero `.idx`

<sup>5</sup>En sistemas que no soportan nombres de ficheros mayores de 8 caracteres, el programa puede llamarse `makeidx`.

puede procesarse con el programa `makeindex`.

`makeindex nombrefichero`

El programa `makeindex` genera un índice ordenado con el mismo nombre base, pero esta vez con la extensión `.ind`. Si se vuelve a procesar el fichero de entrada  $\text{\LaTeX}$ , este índice ordenado se incluye en el documento en el punto donde  $\text{\LaTeX}$  encuentra la orden

`\printindex`

El paquete `showidx` que viene con  $\text{\LaTeX} 2_{\epsilon}$  imprime todas las entradas del índice en el margen izquierdo del texto. Esto es bastante útil para revisar el índice de un documento.

Tenga en cuenta que la orden `\index` puede afectar al aspecto del documento si no se usa con cuidado.

Palabra `\index{Palabra}`. Compare con `Palabra\index{Palabra}`. Mire la posición del punto.

Palabra . Compare con Palabra. Mire la posición del punto.

## 4.4. Cabeceras personalizadas

El paquete `fancyhdr`,<sup>6</sup> escrito por Piet van Oostrum, proporciona órdenes para personalizar las cabeceras y pies de página. Si mira a la parte superior de esta página, verá una posible aplicación de este paquete.

El objetivo de personalizar cabeceras y pies es conseguir que funcionen los nombres de sección y capítulo.  $\text{\LaTeX}$  realiza esto en dos etapas. En la definición de la cabecera y el pie, use las órdenes `\rightmark` y `\leftmark` para representar la sección y el capítulo actual, respectivamente. Los valores de estas dos órdenes se sobrescribirán cada vez que se procese una orden de capítulo o sección.

Para flexibilidad total, la orden `\chapter` y similares no redefinen `\rightmark` y `\leftmark` ellas mismas. Llamen a otra orden (`\chaptermark`, `\sectionmark` o `\subsectionmark`) que a su vez es responsable de redefinir `\rightmark` y `\leftmark`.

Si quiere cambiar el aspecto del nombre del capítulo en la cabecera, necesita solamente “renovar” la orden `\chaptermark`.

La figura 4.1 muestra una configuración posible para el paquete `fancyhdr` que hace que las cabeceras aparezcan como en este libro. En cualquier caso, consulte la documentación del paquete.

---

<sup>6</sup>Disponible en `macros/latex/contrib/supported/fancyhdr`.

---

```
\documentclass{book}
\usepackage{fancyhdr}
\pagestyle{fancy}
% con esto nos aseguramos de que las cabeceras
% de capítulo y de sección vayan en minúsculas
\renewcommand{\chaptermark}[1]{%
    \markboth{#1}{} }
\renewcommand{\sectionmark}[1]{%
    \markright{\thesection\ #1} }
\fancyhf{} % borra cabecera y pie actuales
\fancyhead[LE,R0]{\bfseries\thepage}
\fancyhead[LO]{\bfseries\rightmark}
\fancyhead[RE]{\bfseries\leftmark}
\renewcommand{\headrulewidth}{0.5pt}
\renewcommand{\footrulewidth}{0pt}
\addtolength{\headheight}{0.5pt} % espacio para la raya
\fancypagestyle{plain}{%
    \fancyhead{} % elimina cabeceras en páginas "plain"
    \renewcommand{\headrulewidth}{0pt} % así como la raya
}
```

---

Figura 4.1: Ejemplo de configuración de fancyhdr.

## 4.5. El paquete Verbatim

Ya hemos hablado antes del *entorno* `verbatim`. En esta sección, va a conocer el *paquete* `verbatim`. El paquete `verbatim` es básicamente una re-implementación del entorno `verbatim` que soslaya algunas de las limitaciones del entorno `verbatim` original. Esto no es espectacular en sí mismo, pero la implementación del paquete `verbatim` añade nuevas prestaciones, que es por lo que menciono aquí el paquete. El paquete `verbatim` proporciona la orden

```
\verbatiminput{nombrefichero}
```

que le permite incluir un fichero de texto en su documento como si estuviera dentro de un entorno `verbatim`.

Como el paquete `verbatim` es parte del lote ‘tools’, debería encontrarse pre-instalado en la mayoría de los sistemas. Si quiere saber más sobre este paquete, lea [10].

## 4.6. Instalación de paquetes adicionales

La mayoría de las instalaciones  $\text{\LaTeX}$  vienen con un amplio conjunto de paquetes de estilo pre-instalados, pero muchos más están disponibles en la red. El sitio principal para buscarlos es CTAN (<http://www.ctan.org/>).

Los paquetes como `geometry`, `hyphenat` y muchos otros consisten habitualmente en dos ficheros: un fichero con extensión `.ins` y otro con extensión `.dtx`. Suele haber un fichero `readme.txt` con una breve descripción del paquete. Debería leer ese fichero en primer lugar, por supuesto.

En cualquier caso, una vez haya copiado los ficheros del paquete en su máquina, todavía tendrá que procesarlos de modo que (a) informe a su distribución  $\text{\TeX}$  sobre el nuevo paquete y (b) le dé la documentación. He aquí cómo puede hacer la primera parte:

1. Ejecute  $\text{\LaTeX}$  con el fichero `.ins`. Esto extraerá un fichero `.sty`.
2. Mueva el fichero `.sty` a un lugar donde su distribución pueda encontrarlo. Suele ser en el subdirectorio `.../texmf/tex/latex` (los usuarios de ReactOS o Windows deben cambiar la inclinación de las barras).
3. Refresque la base de datos de nombres de fichero de su distribución. La orden depende de la distribución de  $\text{\LaTeX}$  que use: `teTeX`, `fpTeX` – `texhash`; `web2c` – `maktexlsr`; `MikTeX` – `initexmf -update-fndb` o use la interfaz gráfica.

Ahora puede extraer la documentación del fichero `.dtx`:

1. Ejecute L<sup>A</sup>T<sub>E</sub>X con el fichero `.dtx`. Esto generará un fichero `.dvi`. Quizá tenga que ejecutar L<sup>A</sup>T<sub>E</sub>X varias veces antes de que las referencias cruzadas se establezcan correctamente.
2. Compruebe si L<sup>A</sup>T<sub>E</sub>X ha producido un fichero `.idx` entre los ficheros que tiene ahora. Si no ve este fichero, avance al paso 5.
3. Para generar el índice, escriba lo siguiente:

`makeindex -s gind.ist nombre`

(donde *nombre* es el nombre del fichero principal sin extensión).
4. Ejecute L<sup>A</sup>T<sub>E</sub>X con el fichero `.dtx` otra vez.
5. Finalmente, haga un fichero `.ps` o `.pdf` para imprimir cómodamente.

A veces verá que se ha creado un fichero `.glo` (glosario). Ejecute las siguientes órdenes entre los pasos 4 y 5:

```
makeindex -s gglo.ist -o nombre.gls nombre.glo
```

Asegúrese de ejecutar L<sup>A</sup>T<sub>E</sub>X con el `.dtx` una última vez antes de proceder al paso 5.

## 4.7. Uso de pdfL<sup>A</sup>T<sub>E</sub>X

Por Daniel Flipo <Daniel.Flipo@univ-lille1.fr>

PDF es un formato de documento de tipo hipertexto. Como en una página web, algunas palabras del documento se marcan como hiperenlaces. Enlazan a otros lugares del documento o incluso a otros documentos. Si pica en un hiperenlace se traslada al destino del enlace. En el contexto de L<sup>A</sup>T<sub>E</sub>X, esto significa que todas las apariciones de `\ref` y `\pageref` se vuelven hiperenlaces. Además, el índice general, el índice alfabético y otras estructuras similares se convierten en colecciones de hiperenlaces.

La mayoría de páginas web que encuentra hoy se escriben en HTML (*HyperText Markup Language*). Este formato tiene dos desventajas significativas a la hora de escribir documentos científicos:

1. No hay un procedimiento general para incluir fórmulas matemáticas en documentos HTML. Aunque hay una norma MathML para ello, la mayoría de los navegadores todavía no la soportan, o carecen de las fundiciones adecuadas.
2. Imprimir documentos HTML es posible, pero los resultados varían mucho entre plataformas y navegadores. Los resultados son mucho peores de lo que esperaríamos de L<sup>A</sup>T<sub>E</sub>X.

Ha habido muchos intentos de crear traductores de L<sup>A</sup>T<sub>E</sub>X a HTML. Algunos han tenido incluso bastante éxito en el sentido de que pueden producir



páginas web legibles a partir de un fichero de entrada  $\text{\LaTeX}$  normal. Pero todos ellos prescinden de ciertos detalles para conseguir hacer su trabajo. En cuanto uno comienza a usar funciones más complejas de  $\text{\LaTeX}$  y paquetes externos, las cosas tienden a desplomarse. Los autores que deseen preservar la calidad tipográfica única de sus documentos incluso al publicarlos en la web deben considerar usar PDF (*Portable Document Format*), que preserva el aspecto del documento y permite navegación hipertextual. Muchos navegadores modernos tienen extensiones que permiten mostrar directamente los documentos PDF.

Aunque hay visores DVI y PS para casi todas las plataformas, verá que los visores PDF como Acrobat Reader y Xpdf están incluso más extendidos. Así que distribuir versiones PDF de sus documentos los hará mucho más accesibles a sus lectores potenciales.

#### 4.7.1. Documentos PDF para la red

La creación de un fichero PDF de fuente  $\text{\LaTeX}$  es muy simple, gracias al programa  $\text{pdf}\text{\TeX}$  desarrollado por Hàn Thế Thành.  $\text{pdf}\text{\TeX}$  produce salida PDF donde el  $\text{\TeX}$  normal produce DVI. Existe también  $\text{pdf}\text{\LaTeX}$ , que produce salida PDF de fuentes  $\text{\LaTeX}$ .

Tanto  $\text{pdf}\text{\TeX}$  como  $\text{pdf}\text{\LaTeX}$  se instalan automáticamente en muchas distribuciones  $\text{\TeX}$  modernas, como  $\text{te}\text{\TeX}$ ,  $\text{fp}\text{\TeX}$ ,  $\text{Mik}\text{\TeX}$ ,  $\text{\TeX}$ Live y  $\text{CMac}\text{\TeX}$ .

Para producir un PDF en lugar de un DVI, es suficiente remplazar la orden `latex fichero.tex` por `pdflatex fichero.tex`. En sistemas donde  $\text{\LaTeX}$  no se llama desde una línea de órdenes, ha de haber un botón especial en la barra de herramientas  $\text{\TeX}$ .

En  $\text{\LaTeX}$  puede definir el tamaño del papel con un argumento opcional de documentclass como `a4paper` o `letterpaper`. Esto funciona en  $\text{pdf}\text{\LaTeX}$ , pero además de esto  $\text{pdf}\text{\TeX}$  también necesita saber el tamaño físico del papel para determinar el tamaño físico de las páginas en el fichero pdf. Si usa el paquete `hyperref` (véase página 85), el tamaño del papel se ajustará automáticamente. Si no, tendrá que hacerlo manualmente poniendo las siguientes líneas en el preámbulo del documento:

```
\pdfpagewidth=\paperwidth
\pdfpageheight=\paperheight
```

La sección siguiente dará más detalles de las diferencias entre  $\text{\LaTeX}$  normal y  $\text{pdf}\text{\LaTeX}$ . Las principales diferencias conciernen a tres asuntos: las fundiciones, el formato de las imágenes por incluir y la configuración manual de hiperenlaces.

### 4.7.2. Las fundiciones

pdfL<sup>A</sup>T<sub>E</sub>X puede trabajar con todo tipo de fundiciones (PK bitmaps, TrueType, POSTSCRIPT type 1...) salvo el formato de fundición normal L<sup>A</sup>T<sub>E</sub>X; las fundiciones bitmap PK producen resultados muy feos cuando el documento se muestra con Acrobat Reader. Es mejor usar fundiciones POSTSCRIPT Type 1 exclusivamente para producir documentos que aparezcan bien. *Las instalaciones TeX modernas se configurarán automáticamente para que esto ocurra. Lo mejor es probarlo. Si funciona para usted, simplemente sáltese esta sección.*

La implementación POSTSCRIPT Type 1 de las fundiciones Computer Modern y AMSFonts fue producida por Blue Sky Research y Y&Y, Inc., que transfirieron los derechos de copia a la American Mathematical Society. Las fundiciones se hicieron libres en 1997 y actualmente vienen con casi todas las distribuciones T<sub>E</sub>X.

Sin embargo, si desea crear con L<sup>A</sup>T<sub>E</sub>X documentos otros idiomas aparte del inglés, podría querer usar las fundiciones EC, LH o CB. Vladimir Volovich ha creado el lote de fundiciones cm-super que cubre todos los conjuntos de fundiciones EC/TC, EC Concrete, EC Bright y LH. Está disponible en `CTAN:/fonts/ps-type1/cm-super` y se incluye en T<sub>E</sub>XLive7 y MikT<sub>E</sub>X. Otras fundiciones parecidas type 1 CB griegas creadas por Apostolos Syropoulos están disponibles en `CTAN:/tex-archive/fonts/greek/cb`. Lamentablemente, ninguna de ellas tiene la misma calidad tipográfica que las fundiciones Type1 CM de Blue Sky/Y&Y. Fueron pergeñadas automáticamente, y el documento podría no parecer tan claro en la pantalla como los que usan fundiciones Blue Sky/Y&Y type 1 CM; en dispositivos de salida de alta resolución producen idéntico resultado a las fuentes originales bitmap EC/LH/CB.

Si crea documentos en un lenguaje con alfabeto latino, tiene otras opciones.

- Podría usar el paquete `aeguill`, alias *Almost European Computer Modern with Guillemets*. Basta con que ponga el renglón `\usepackage{aeguill}` en el preámbulo de su documento para habilitar las fundiciones AE virtuales en lugar de las fundiciones EC.
- Puede usar el paquete `mltex`, pero solamente funciona si su pdfT<sub>E</sub>X ha sido compilado con la opción `mltex`.

Las fundiciones AE virtuales, como el sistema M<sup>I</sup>T<sub>E</sub>X, hacen que T<sub>E</sub>X crea que tiene una fundición completa de 256 caracteres a su disposición creando casi todos los caracteres ausentes a partir de caracteres de la fundición CM y reordenándolos en el orden EC; esto permite usar las excelentes fundiciones CM de formato type 1 disponibles en muchos sistemas. Como la fundición tiene ahora una codificación T1, la silabación funcionará bien en idiomas

Europeos con alfabeto latino. La única desventaja de este enfoque es que los caracteres artificiales AE no funcionan con la función `Find` de Acrobat Reader, así que no puede buscar palabras con acentos en su fichero PDF final si usa ese visor.

Para el idioma ruso una solución similar es usar las fundiciones virtuales C1 disponibles en `ftp://ftp.vsu.ru/pub/tex/font-packs/c1fonts`. Estas fundiciones combinan las fundiciones normales CM type 1 de la colección Bluesky y las fundiciones CMCYR type 1 de la colección Paradissa y BaKo-Ma, todas disponibles en CTAN. Las fundiciones Paradissa contienen sólo letras rusas; las fundiciones C1 carecen de otros caracteres cirílicos.

Otra solución es cambiar a otras fundiciones POSTSCRIPT type 1. De hecho, algunas de ellas incluso están incluidas con cada copia de Acrobat Reader. Ya que estas fundiciones tienen diferentes tamaños de carácter, la composición del texto en sus páginas cambiará. Generalmente estas otras fundiciones usarán más espacio que las fundiciones CM, que son más eficientes. También, la coherencia global visual de su documento se resentirá porque Times, Helvetica y Courier (los candidatos primeros para tal sustitución) no han sido diseñadas para quedar en armonía en el mismo documento.

Dos conjuntos de fundiciones preparados y disponibles para este propósito: `pxfonts`, que está basado en *Palatino* como su principal fundición para el cuerpo del texto, y el paquete `txfonts`, que está basado en *Times*. Para usarlos basta con poner las siguientes líneas en el preámbulo de su documento:

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

Nota: puede hallar líneas como

**Warning: pdftex (file eurmo10): Font eur... not found**

en el fichero `.log` tras compilar su fichero de entrada. Significan que algunos tipos usadas en el documento no han sido encontradas. Debería resolver estos problemas, pues de lo contrario el documento PDF resultante puede *no mostrar las páginas con los caracteres que faltan*.

Como puede ver, este asunto sobre fundiciones, especialmente la falta de un buen conjunto de fundiciones EC equivalente en calidad a la fundición CM en formato type 1, ha ocupado la mente de mucha gente. Hace poco se ha anunciado la disponibilidad de un nuevo conjunto de fuentes vectoriales de alta calidad llamado Latin Modern (LM). Es el fin de la miseria. Si tiene una instalación T<sub>E</sub>X reciente, tiene muchas posibilidades de tener una copia instalada; todo lo que necesita hacer es añadir

```
\usepackage{lmodern}
\usepackage[T1]{fontenc}
\usepackage{textcomp}
```

al preámbulo de su documento y está usted listo para crear excelentes salidas pdf con soporte completo de todo el conjunto de caracteres latinos.

### 4.7.3. Uso de gráficos

Incluir gráficos en un documento funciona mejor con el paquete `graphicx` (véase pág. 73). Usando la opción del *controlador* especial `pdftex` el paquete trabajará también con pdfL<sup>A</sup>T<sub>E</sub>X:

```
\usepackage[pdftex]{color,graphicx}
```

En el código he incluido el opción `color`, pues es natural usar color en documentos expuestos en la red.

Hasta ahora todo buenas noticias. Las malas noticias son que los gráficos en formato Encapsulated POSTSCRIPT no funcionan con pdfL<sup>A</sup>T<sub>E</sub>X. Si no incluye una extensión de fichero en la orden `\includegraphics`, `graphicx` buscará un fichero adecuado, en función de lo establecido en la opción del *controlador*. Para `pdftex` esto significa los formatos `.png`, `.pdf`, `.jpg` y `.mps` (METAPOST), pero *no* `.eps`.

La única salida a este problema es convertir los ficheros EPS al formato PDF usando la utilidad `epstopdf` disponible en muchos sistemas. Para gráficos vectoriales (dibujos) esto es una buena solución. Para gráficos pixelados (fotos, escaneados) no es ideal, porque el formato PDF soporta nativamente la inclusión de imágenes PNG y JPEG. PNG es bueno para capturas de pantalla y otras imágenes con pocos colores, y admite transparencia. JPEG es bueno para fotos, porque ahorra mucho espacio.

Incluso puede ser deseable no dibujar ciertas figuras geométricas, sino describirlas mediante un lenguaje especializado, como METAPOST, que puede encontrarse en muchas distribuciones T<sub>E</sub>X, y viene con su propio manual exhaustivo.

### 4.7.4. Enlaces de hipertexto

El paquete `hyperref` se ocupará de convertir todas las referencias internas de su documento en hiperenlaces. Para que esto funcione automáticamente se requiere algo de magia, así que tendrá que poner `\usepackage[pdftex]{hyperref}` como la *última* orden en el preámbulo de su documento.

Para controlar el comportamiento del paquete `hyperref` se dispone de muchas opciones:

- o como una lista separada por comas tras la opción `pdftex`  
`\usepackage[pdftex]{hyperref}`
- o en líneas individuales con la orden `\hypersetup{opciones}`.

La única opción requerida es `pdftex`; las otras son opcionales y permiten cambiar el comportamiento por omisión de `hyperref`.<sup>7</sup> El la siguiente lista los valores por omisión se escriben con una fundición recta.

---

<sup>7</sup>Vale la pena comentar que este paquete no se limita a trabajar con pdfT<sub>E</sub>X. Puede

**bookmarks** (**=true,false**) muestra u oculta la barra de marcadores al representar el documento

**unicode** (**=false,true**) permite usar caracteres de alfabetos no latinos en los marcadores

**pdftoolbar** (**=true,false**) muestra u oculta la barra de herramientas

**pdfmenubar** (**=true,false**) muestra u oculta la barra de menús

**pdffitwindow** (**=true,false**) ajusta el tamaño del documento mostrado a la ventana del visor

**pdftitle** (**=`{texto}`**) define el título que se muestra en la ventana del visor Document Info

**pdfauthor** (**=`{text}`**) el nombre del autor del PDF

**pdfnewwindow** (**=true,false**) define si debe abrirse una nueva ventana cuando un enlace apunta fuera del documento actual

**colorlinks** (**=false,true**) rodea los enlaces con marcos de color (**false**) o colorea el texto de los enlaces (**true**); el color de los enlaces se configura con las siguientes opciones (se muestran los colores por omisión):

**linkcolor** (**=red**) color de enlaces internos (secciones, páginas, etc.),

**citecolor** (**=green**) color de enlaces de cita (bibliografía)

**filecolor** (**=magenta**) color de enlaces a ficheros

**urlcolor** (**=cyan**) color de enlaces a la red (HTTP, FTP, correo electrónico)

Si está contento con los valores por omisión, use simplemente

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

Para tener abierta la lista de marcadores y en color los enlaces (los valores **=true** son opcionales):

```
\usepackage[pdftex,bookmarks,colorlinks]{hyperref}
```

Al crear PDFs destinados a la impresión, los enlaces coloreados no son buenos pues acaban siendo grises (y, por tanto, difíciles de leer) en la salida final. Puede usar cuadros de color, que no se imprimen:

---

configurarse para empotrar información específica de PDF en la salida DVI del L<sup>A</sup>T<sub>E</sub>X normal, que después se pasa al fichero PS mediante **dvips** y que finalmente se integra en el PDF. No es lo más eficiente, pero es posible.

```
\usepackage{hyperref}
\hypersetup{colorlinks=false}
```

o hacer negros los enlaces:

```
\usepackage{hyperref}
\hypersetup{colorlinks,%
            citecolor=black,%
            filecolor=black,%
            linkcolor=black,%
            urlcolor=black,%
            pdftex}
```

Cuando quiera proporcionar información para la sección Document Info del fichero PDF:

```
\usepackage[pdfauthor={Ludoviko Lazaro Zamenhof},%
            pdftitle={Esperanto: lingvo internacia},%
            pdftex]{hyperref}
```

Además de los hiperenlaces automáticos para referencias cruzadas, es posible empotrar enlaces explícitos usando

`\href{destino}{texto}`

El código

```
El sitio de \href{http://www.ctan.org}{CTAN}.
```

produce la salida “CTAN”; picando en la palabra “CTAN” le conducirá al sitio web de CTAN.

Si el destino del enlace es un fichero local, puede usar la orden `\href`:

```
El documento completo está \href{manual.pdf}{aquí}
```

que produce el texto “El documento completo está aquí”. Picando en la palabra “aquí” abrirá el fichero `manual.pdf`. (El nombre de fichero es relativo a la situación del documento actual).

El autor de un artículo puede querer que sus lectores le envíen fácilmente mensajes electrónicos usando la orden `\href` dentro de la orden `\author` en la página del título del documento:

```
\author{Mary Oetiker $<\href{mailto:mary@oetiker.ch}%
        {mary@oetiker.ch}$>$}
```

Fíjese en que he puesto el enlace de forma que mi dirección electrónica aparece no sólo en el enlace sino también en la misma página. Lo hice así porque el enlace

```
\href{mailto:mary@oetiker.ch}{Mary Oetiker}
```

funcionaría bien en el visor de PDF, pero una vez impresa la página ya no se podría ver la dirección electrónica.

#### 4.7.5. Problemas con enlaces

Mensajes como el siguiente:

```
! pdfTeX warning (ext4): destination with the same
  identifier (name{page.1}) has been already used,
  duplicate ignored
```

aparecen cuando un contador se reinicializa, por ejemplo al usar la orden `\mainmatter` proporcionada por la clase de documento `book`. Restaura el contador de número de página a 1 antes del primer capítulo del libro. Pero como el prefacio del libro también tiene una página número 1 todos los enlaces a la “page 1” ya no serán únicos, de ahí la advertencia “`duplicate has been ignored`.”

El antídoto consiste en poner `plainpages=false` en las opciones de `hyperref`. Lamentablemente eso sólo funciona con el contador de páginas. Una medida más radical es usar la opción `hypertextnames=false`, pero ocasiona que los enlaces a página en el índice dejen de funcionar.

#### 4.7.6. Problemas con marcadores

El texto mostrado por los marcadores no siempre aparece como usted pretendía. Puesto que los marcadores son “sólo texto”, se dispone de muchos menos caracteres para los marcadores que para el texto  $\text{\LaTeX}$  normal. `Hyperref` normalmente se dará cuenta de tales problemas y advertirá:

```
Package hyperref Warning:
Token not allowed in a PDFDocEncoded string:
```

Puede soslayar este problema proporcionando una cadena de texto para los marcadores, que remplace el texto ofensivo:

```
\texorpdfstring{texto  $\text{\TeX}$ }{Texto marcador}
```

Las expresiones de mates son un candidato idóneo para estos problemas:

```
\section{\texorpdfstring{$E=mc^2$}%
{E=mc^2}}
```

que convierte `\section{$E=mc^2$}` a “E=mc<sup>2</sup>” en el área del marcador.

Los cambios de color tampoco van bien en los marcadores:

```
\section{\textcolor{red}{Red !}}
```

produce la cadena “redRed!”. La orden `\textcolor` no es tenida en cuenta pero su argumento se imprime.

Si usa

```
\section{\texorpdfstring{\textcolor{red}{Red !}}{Red\ !}}
```

el resultado será mucho más legible.

Si escribe un documento en unicode y usa la opción `unicode` para el paquete `hyperref` puede usar caracteres unicode en los marcadores. Esto le dará una selección mucho mayor de caracteres cuando use `\texorpdfstring`.

### Compatibilidad de fuente entre L<sup>A</sup>T<sub>E</sub>X y pdfL<sup>A</sup>T<sub>E</sub>X

Lo ideal sería que su documento compilase igual de bien con L<sup>A</sup>T<sub>E</sub>X y pdfL<sup>A</sup>T<sub>E</sub>X. El principal problema al respecto es la inclusión de los gráficos. La solución simple es *omitir sistemáticamente* la extensión de fichero de las órdenes `\includegraphics`. Así buscarán automáticamente un fichero del formato adecuado en el directorio actual. Todo lo que ha de hacer es crear versiones apropiadas de los ficheros gráficos. L<sup>A</sup>T<sub>E</sub>X buscará `.eps`, y pdfL<sup>A</sup>T<sub>E</sub>X intentará incluir un fichero con la extensión `.png`, `.pdf`, `.jpg` o `.mps` (en ese orden).

Para los casos en que quiera usar código diferente para la versión PDF de su documento, puede simplemente añadir el paquete `ifpdf`<sup>8</sup> en su preámbulo. Es muy posible que ya lo tenga instalado; si no, quizás esté usando MiK<sub>T</sub>E<sub>X</sub> que se lo instalará automáticamente la primera vez que trate de usarlo. Este paquete define la orden especial `\ifpdf` que le permitirá escribir código condicional fácilmente. En este ejemplo, queremos que la versión PostScript sea en blanco y negro por los costos de impresión pero queremos que la versión PDF para consultar en la red sea en color.

```
\RequirePackage{ifpdf} % ¿ejecutar con pdfTeX?
\ifpdf
  \documentclass[a4paper,12pt,pdftex]{book}
\else
  \documentclass[a4paper,12pt,dvips]{book}
\fi

\ifpdf
  \usepackage{lmodern}
```

---

<sup>8</sup>Si quiere conocer la historia completa de este paquete, vaya a la T<sub>E</sub>X FAQ bajo el epígrafe <http://www.tex.ac.uk/cgi-bin/texfaq2html?label=ifpdf>.



```

\fi
\usepackage[bookmarks, % añadir hiperenlaces
            colorlinks,
            plainpages=false]{hyperref}
\usepackage[T1]{fontenc}
\usepackage[latin1]{inputenc}
\usepackage[english]{babel}
\usepackage{graphicx}
...

```

En este ejemplo he incluido el paquete `hyperref` incluso en la versión no-PDF. El efecto de esto es hacer que la orden `\href` funcione en todos los casos, lo que me ahorra cubrir cada aparición en una sentencia condicional.

Tenga en cuenta que las distribuciones  $\text{\TeX}$  recientes ( $\text{\TeX}$ Live por ejemplo), el programa normal  $\text{\TeX}$  es realmente `pdf $\text{\TeX}$`  que cambia automáticamente entre producir pdf o dvi según la configuración de la clase de documento. Si usa el código de arriba entonces todavía puede usar la orden `pdflatex` para conseguir salida pdf y `latex` para salida DVI normal.

## 4.8. Trabajo con $\text{\XeLaTeX}$

Por Axel Kielhorn <[A.Kielhorn@web.de](mailto:A.Kielhorn@web.de)>

La mayoría de las cosas dichas sobre `pdf $\text{\LaTeX}$`  son también válidas para  $\text{\XeLaTeX}$ .

Hay una wiki en <http://wiki.xelatex.org/doku.php> que recoge información relevante sobre  $\text{\XeTeX}$  y  $\text{\XeLaTeX}$ .

### 4.8.1. Las fundiciones

Además de las fundiciones basadas en las `tfm` normales,  $\text{\XeLaTeX}$  es capaz de usar cualquier fundición conocida por el sistema operativo. Si tiene la fundición `Linux Libertine` instalada, simplemente dice

```

\usepackage{fontspec}
\setmainfont[Ligatures=TeX]{Linux Libertine}

```

en el preámbulo. Esto normalmente detectará también las versiones cursiva y negrita, así que `\textit` y `\textbf` funcionarán como de costumbre. Cuando la fundición usa tecnología OpenType tendrá acceso a muchas capacidades que en el pasado requerían cambio a una fundición distinta o el uso de fuentes virtuales. La característica principal es el conjunto de caracteres extendido; una fundición puede contener caracteres latinos, griegos y cirílicos y sus correspondientes ligaduras.

Muchas fundiciones contienen al menos dos tipos de cifras numerales, las “normales” y las llamadas “antiguas” (*Old Style*, o “bajas”), que se extienden

bajo la línea del renglón. Pueden contener cifras proporcionales (el “1” lleva menos espacio que el “0”) o cifras a máquina (*monospaced*), que son más apropiadas para cuadros y tablas.

```
\newfontfamily\LLln[Numbers=Lining]{(font)}
\newfontfamily\LLos[Numbers=OldStyle]{(font)}
\newfontfamily\LLlnm[Numbers=Lining,Numbers=Monospaced]{(font)}
\newfontfamily\LLosm[Numbers=OldStyle,Numbers=Monospaced]{(font)}
```

Casi todas las fundiciones OpenType contienen las ligaduras habituales (fl fi ffi) pero también hay otras ligaduras raras o históricas, como st, ct y tz. Puede que no quiera usarlas en un reporte técnico, pero están bien en una novela. Puede activarlas usando cualquiera de las siguientes líneas:

```
\setmainfont[Ligatures=Rare]{(font)}
\setmainfont[Ligatures=Historic]{(font)}
\setmainfont[Ligatures=Historic,Ligature=Rare]{(font)}
```

No toda fundición contiene ambos conjuntos de ligaduras; consulte la documentación o simplemente inténtelo. Algunas veces esas ligaduras dependen del idioma; por ejemplo, una ligadura usada en polaco (fk) no se usa en inglés. Debe añadir

```
\setmainfont[Language=Polish]{(font)}
```

para activar las ligaduras polacas.

Algunas fundiciones (como la comercial Adobe Garamond Premier Pro) contienen caracteres alternos que se activan por omisión en X<sub>Y</sub>L<sup>A</sup>T<sub>E</sub>X distribuido con T<sub>E</sub>XLive 2010.<sup>9</sup> El resultado es una “Q” estilizada, con una curva debajo de la letra siguiente, si es una “u”. Para inhabilitar esta característica debe definir la fundición con las estilizaciones contextuales desactivadas:

```
\setmainfont[Contextuals=NoAlternate]{(font)}
```

Para más información sobre fundiciones en X<sub>Y</sub>L<sup>A</sup>T<sub>E</sub>X lea el manual de fontspec.

### ¿Dónde consigo fundiciones OpenType?

Si tiene TeXLive instalado, ya tiene algunas en `../texmf-dist/fonts/opentype`, sólo instálelas en su sistema operativo. Esta colección no incluye DejaVu, que está disponible en <http://dejavu-fonts.org/>.

Cerciórese de que cada fundición está instalada *sólo una vez*, de lo contrario puede obtener resultados interesantes.

---

<sup>9</sup>El comportamiento ha cambiado con la versión 2010; estaba desactivado en versiones anteriores.

Puede usar todas las fundiciones instaladas en su ordenador, pero recuerde que otros usuarios pueden no tenerlas. La fundición Zapfino usada en el manual de `fontspec` está incluida en Mac OSX, pero no está disponible en Windows.<sup>10</sup>

### Uso de caracteres Unicode

El número de caracteres en una fundición ha crecido, pero el número de teclas no. Si esto es así, ¿cómo introducir caracteres no disponibles en el teclado?

Si escribe una gran cantidad de texto en un idioma extranjero, puede instalar un teclado para ese idioma e imprimir las posiciones de los caracteres. (La mayoría de los sistemas operativos tiene una especie de teclado virtual: sólo imprima una captura de pantalla.)

Si usa caracteres exóticos raramente, puede escogerlo directamente en el menú de caracteres.

Algunos entornos (por ejemplo, el sistema X-Windows) ofrecen muchos métodos para introducir caracteres. Algunos editores de texto (por ejemplo, Vim o Emacs) ofrecen formas de introducirlos. Lea el manual de las utilerías que utiliza.

#### 4.8.2. Compatibilidad entre $\text{X}\text{\LaTeX}$ y $\text{pdf}\text{\LaTeX}$

Hay algunas cosas que son diferentes entre  $\text{X}\text{\LaTeX}$  y  $\text{pdf}\text{\LaTeX}$ .

- Un documento de  $\text{X}\text{\LaTeX}$  debe ser escrito en Unicode (UTF-8), mientras  $\text{pdf}\text{\LaTeX}$  puede usar distintas codificaciones de entrada.
- Los paquetes `microtype` todavía no funcionan con  $\text{X}\text{\LaTeX}$ , soporte para caracteres salientes está todavía en desarrollo.
- Cualquier cosa relacionada con fundiciones debe ser revisada. (A menos que quiera limitarse a Latin Modern.)

### 4.9. Creación de presentaciones

Por Daniel Flipo <Daniel.Flipo@univ-lille1.fr>

Puede presentar los resultados de su trabajo científico en un encerado, con transparencias o directamente desde su ordenador portátil utilizando un programa de presentaciones.

$\text{pdf}\text{\LaTeX}$  combinado con la clase `beamer` le permite crear presentaciones en PDF con un aspecto muy similar al que podría obtener con OpenOffice Impress, y con un resultado mucho más portable que si usara PowerPoint, pues los visores de PDF están disponibles en muchos más sistemas.

---

<sup>10</sup>Una versión comercial de esta fundición llamada Zapfino Extra está disponible.

La clase `beamer` usa `graphicx`, `color` y `hyperref` con opciones adaptadas a presentaciones en pantalla.

Cuando compile el código presentado en la figura 4.2 con PDF $\text{\LaTeX}$  obtendrá un fichero PDF con una página de título y una segunda página que muestra varios elementos que serán mostrados uno de cada vez según avance la presentación.

Una de las ventajas de la clase `beamer` es que produce un fichero PDF que es directamente usable sin tener que pasar primero por PostScript, como hace `prosper`, y sin requerir un postprocesamiento adicional como las presentaciones creadas con el paquete `ppower4`.

Con la clase `beamer` puede producir varias versiones (modos) de su documento a partir del mismo fichero de entrada. El fichero de entrada puede contener instrucciones especiales para los diferentes modos entre ángulos. Se dispone de los siguientes modos.

**beamer** para la presentación PDF comentada arriba.

**trans** para transparencias.

**handout** para la versión impresa.

El modo por omisión es `beamer`; puede cambiarlo poniendo un modo diferente como opción global, como `\documentclass[10pt,handout]{beamer}` para imprimir, por ejemplo.

El aspecto de la presentación en pantalla depende del tema que escoja. Puede o escoger uno de los temas distribuidos con la clase `beamer` o crear uno. Vea la documentación de la clase `beamer` en `beameruserguide.pdf` para más información sobre esto.

Echemos un vistazo más cerca al código de la figura 4.2.

Para la versión de pantalla `\mode<beamer>` hemos escogido el tema *Goettingen* que muestra un panel de navegación integrado en el índice general. Las opciones permiten escoger el tamaño del panel (22 mm en este caso) y su posición (a la derecha del texto). La opción *hideothersubsections* muestra los títulos de los capítulos, pero sólo las subsecciones del capítulo actual. No hay configuración especial para `\mode<trans>` y `\mode<handout>`. Aparecen con su aspecto por omisión.

Las órdenes `\title{}`, `\author{}`, `\institute{}` y `\titlegraphic{}` establecen el contenido de la página de título. Los argumentos opcionales de `\title[]{}{}` y `\author[]{}{}` le dejan indicar una versión especial del título y el nombre del autor que se mostrará en el panel del tema *Goettingen*.

Los títulos y subtítulos del panel se crean con órdenes `\section{}` y `\subsection{}` normales que usted coloca *fuera* del entorno `frame`.

Los pequeños iconos de navegación abajo en la pantalla también permiten navegar por el documento. Su presencia es independiente del tema escogido.

```
\documentclass[10pt]{beamer}
\mode<beamer>{%
  \usetheme[hideothersubsections,
            right,width=22mm]{Goettingen}
}

\title{Presentación simple}
\author[D. Flipo]{Daniel Flipo}
\institute{U.S.T.L. \& GUTenberg}
\titlegraphic{\includegraphics[width=20mm]{USTL}}
\date{2005}

\begin{document}

\begin{frame}<handout:0>
  \titlepage
\end{frame}

\section{Un ejemplo}

\begin{frame}
  \frametitle{Cosas por hacer un domingo por la tarde}
  \begin{block}{Uno podría...}
    \begin{itemize}
      \item pasear el perro... \pause
      \item leer un libro\pause
      \item incordiar a un gato\pause
    \end{itemize}
  \end{block}
  y muchas otras cosas
\end{frame}
\end{document}
```

Figura 4.2: Código de ejemplo para la clase beamer

Los contenidos de cada transparencia o pantalla deben colocarse dentro de un entorno `frame`. Hay un argumento opcional entre ángulos (`<` y `>`), que permite suprimir un frame particular en una de las versiones de la presentación. En el ejemplo la primera página no se mostraría en la versión impresa debido al argumento `<handout:0>`.

Es muy recomendable establecer un título para cada transparencia distinto del de la transparencia del título. Esto se hace con la orden `\frametitle{}`. Si se necesita un subtítulo puede usar el entorno `block` como se muestra en el ejemplo. Fíjese en que las órdenes de sección `\section{}` y `\subsection{}` no producen salida en la misma transparencia.

La orden `\pause` en el entorno `itemize` le permite desvelar los puntos uno por uno. Para otros efectos de presentación busque las órdenes `\only`, `\uncover`, `\alt` y `\temporal`. En muchos lugares puede emplear ángulos para personalizar la presentación.

En cualquier caso asegúrese de leer la documentación de la clase `beamer` `beameruserguide.pdf` para disponer de una visión completa de lo que puede ofrecerle. Este paquete está en continuo desarrollo, así que visite su página web <http://latex-beamer.sourceforge.net/> para conseguir la información más actual.



## Capítulo 5

# Producción de gráficos matemáticos

Mucha gente usa  $\text{\LaTeX}$  para componer sus textos; pero además del enfoque orientado a la estructura (y no al contenido) tan conveniente,  $\text{\LaTeX}$  también ofrece la posibilidad (si bien bastante restringida) de producir salidas gráficas a partir de descripciones textuales. Por otro lado, se han creado varias extensiones de  $\text{\LaTeX}$  para evadir estas restricciones. En esta sección aprenderá algunas de ellas.

### 5.1. Panorama general

El entorno `picture` permite programar dibujos directamente en  $\text{\LaTeX}$ . Una descripción detallada puede encontrarse en el  *$\text{\LaTeX}$  Manual* [1]. Por un lado hay restricciones serias, como que las pendientes de los segmentos de recta así como los radios de los círculos están restringidos a un número corto de valores. Por otro lado, el entorno `picture` de  $\text{\LaTeX} 2_{\epsilon}$  trae con él la orden `\qbezier`, donde “q” significa “cuadrática”. Muchas curvas usadas con frecuencia, como círculos, elipses o catenarias, puedes aproximarse satisfactoriamente con curvas de Bézier cuadráticas, aunque esto puede requerir algo de matemáticas. Si además se utiliza un lenguaje de programación como Lisp para generar bloques `\qbezier` de ficheros de entrada  $\text{\LaTeX}$ , el entorno `picture` se vuelve bastante potente.

Aunque la programación de dibujos directamente en  $\text{\LaTeX}$  tiene muchas restricciones, y es a menudo muy incómodo, puede haber razones para hacerlo. Los documentos producidos son “pequeños” en cuanto al tamaño en octetos, y no hay que andar arrastrando ficheros gráficos adicionales.

Los paquetes como `epic` y `eepic` (descritos, por ejemplo, en *The  $\text{\LaTeX}$  Companion* [3]) o `pstricks` ayudan a eliminar las restricciones a las que está sujeto el entorno `picture` original, y refuerzan en gran medida la potencia gráfica de  $\text{\LaTeX}$ .



Mientras los dos primeros paquetes sólo mejoran el entorno `picture`, el paquete `pstricks` tiene su propio entorno de dibujo, `pspicture`. La potencia de `pstricks` se basa en el hecho de que este paquete hace uso extenso de las posibilidades de POSTSCRIPT. Además, numerosos paquetes han sido escritos para propósitos específicos. Uno de ellos es `Xy-pic`, descrito al final de este capítulo. Una amplia variedad de estos paquetes se describe en detalle en *The L<sup>A</sup>T<sub>E</sub>X Graphics Companion* [4] (no lo confunda con *The L<sup>A</sup>T<sub>E</sub>X Companion* [3]).

Quizás la herramienta gráfica más potente relacionada con L<sup>A</sup>T<sub>E</sub>X es `MetaPost`, el gemelo de `METAFONT` de Donald E. Knuth. `MetaPost` tiene el lenguaje de programación de `METAFONT`, muy potente y matemáticamente sofisticado; pero al contrario que `METAFONT`, que genera mapas de píxeles, `MetaPost` genera ficheros de Encapsulated POSTSCRIPT, que pueden importarse en L<sup>A</sup>T<sub>E</sub>X. Para una introducción, vea *A User's Manual for MetaPost* [15], o el tutorial de [17].

Una discusión minuciosa sobre estrategias en L<sup>A</sup>T<sub>E</sub>X y T<sub>E</sub>X para gráficos (y fundiciones) puede encontrarse en *T<sub>E</sub>X Unbound* [16].

## 5.2. El entorno `picture`

Por Urs Oswald <osurs@bluewin.ch>

### 5.2.1. Órdenes básicas

Se crea un entorno `picture`<sup>1</sup> con alguna de las dos órdenes

```
\begin{picture}(x,y)...\end{picture}
```

o

```
\begin{picture}(x,y)(x_0,y_0)...\end{picture}
```

Los números  $x$ ,  $y$ ,  $x_0$ ,  $y_0$  se refieren a `\unitlength`, que puede establecerse en cualquier momento (pero no dentro de un entorno `picture`) con una orden como

```
\setlength{\unitlength}{1.2cm}
```

El valor por omisión de `\unitlength` es `1pt`. El primer par,  $(x, y)$ , reserva dentro del documento un espacio rectangular para el dibujo. El segundo par, opcional,  $(x_0, y_0)$ , asigna coordenadas arbitrarias a la esquina inferior izquierda del rectángulo reservado.

<sup>1</sup>Lo crea o no, el entorno `picture` funciona sin más, con L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> normal, sin necesidad de cargar ningún paquete.

La mayoría de las órdenes de dibujo tienen alguna de las dos formas

`\put (x,y) {objeto}`

o

`\multiput (x,y) (\Delta x, \Delta y) {n} {objeto}`

Las curvas de Bézier son una excepción. Se dibujan con la orden

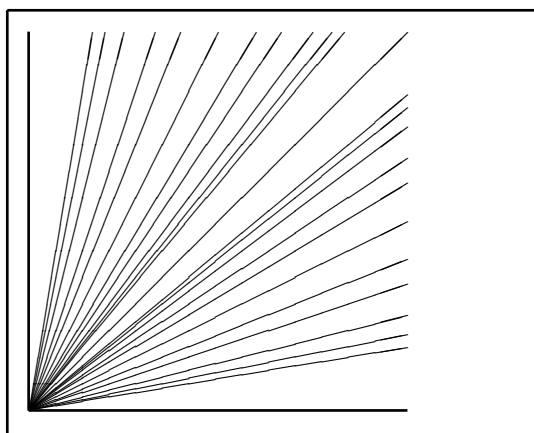
`\qbezier (x_1, y_1) (x_2, y_2) (x_3, y_3)`

### 5.2.2. Segmentos de recta

```

\setlength{\unitlength}{5cm}
\begin{picture}(1,1)
  \put(0,0){\line(0,1){1}}
  \put(0,0){\line(1,0){1}}
  \put(0,0){\line(1,1){1}}
  \put(0,0){\line(1,2){.5}}
  \put(0,0){\line(1,3){.3333}}
  \put(0,0){\line(1,4){.25}}
  \put(0,0){\line(1,5){.2}}
  \put(0,0){\line(1,6){.1667}}
  \put(0,0){\line(2,1){1}}
  \put(0,0){\line(2,3){.6667}}
  \put(0,0){\line(2,5){.4}}
  \put(0,0){\line(3,1){1}}
  \put(0,0){\line(3,2){1}}
  \put(0,0){\line(3,4){.75}}
  \put(0,0){\line(3,5){.6}}
  \put(0,0){\line(4,1){1}}
  \put(0,0){\line(4,3){1}}
  \put(0,0){\line(4,5){.8}}
  \put(0,0){\line(5,1){1}}
  \put(0,0){\line(5,2){1}}
  \put(0,0){\line(5,3){1}}
  \put(0,0){\line(5,4){1}}
  \put(0,0){\line(5,6){.8333}}
  \put(0,0){\line(6,1){1}}
  \put(0,0){\line(6,5){1}}
\end{picture}

```



Se dibujan segmentos de recta con la orden

$\text{\put}(x,y)\{\text{\line}(x_1,y_1)\{length\}$

La orden `\line` tiene dos argumentos:

1. un vector director,
2. una longitud.

Los componentes del vector director están restringidos a los enteros

$$-6, -5, \dots, 5, 6,$$

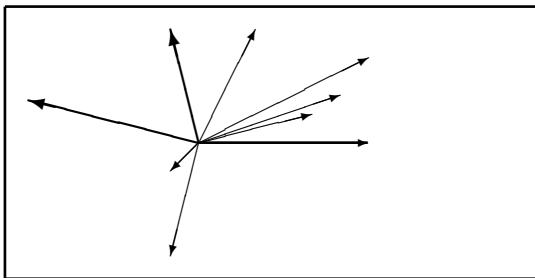
y tienen que ser primos entre sí (coprimos; sin divisor común salvo 1). La figura ilustra los 25 posibles valores de las pendientes en el primer cuadrante. La longitud es relativa a `\unitlength`. El argumento longitud es la coordenada vertical en el caso de un segmento de recta vertical; en el resto de los casos, la coordenada horizontal.

## 5.2.3. Flechas

```

\setlength{\unitlength}{0.75mm}
\begin{picture}(60,40)
  \put(30,20){\vector(1,0){30}}
  \put(30,20){\vector(4,1){20}}
  \put(30,20){\vector(3,1){25}}
  \put(30,20){\vector(2,1){30}}
  \put(30,20){\vector(1,2){10}}
  \thicklines
  \put(30,20){\vector(-4,1){30}}
  \put(30,20){\vector(-1,4){5}}
  \thinlines
  \put(30,20){\vector(-1,-1){5}}
  \put(30,20){\vector(-1,-4){5}}
\end{picture}

```



Las flechas se dibujan con la orden

`\put( $x$ , $y$ ){\vector( $x_1$ , $y_1$ ){\length}}`

Para las flechas, los componentes del vector director están incluso más estrechamente restringidos que para los segmentos de recta, a los enteros

$$-4, -3, \dots, 3, 4$$

Los componentes también tienen que ser primos entre sí (sin divisor común salvo 1). Fíjese en el efecto de la orden `\thicklines` en las dos flechas que apuntan arriba a la izquierda.

## 5.2.4. Circunferencias y círculos

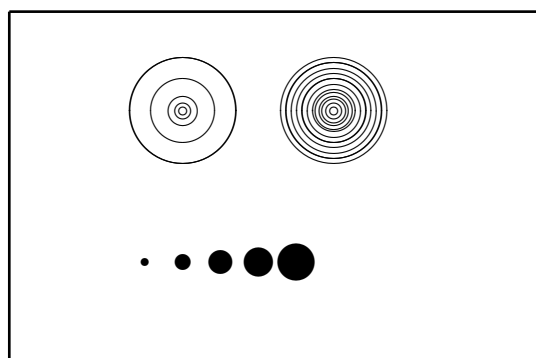
```

\setlength{\unitlength}{1mm}
\begin{picture}(60, 40)
  \put(20,30){\circle{1}}
  \put(20,30){\circle{2}}
  \put(20,30){\circle{4}}
  \put(20,30){\circle{8}}
  \put(20,30){\circle{16}}
  \put(20,30){\circle{32}}

  \put(40,30){\circle{1}}
  \put(40,30){\circle{2}}
  \put(40,30){\circle{3}}
  \put(40,30){\circle{4}}
  \put(40,30){\circle{5}}
  \put(40,30){\circle{6}}
  \put(40,30){\circle{7}}
  \put(40,30){\circle{8}}
  \put(40,30){\circle{9}}
  \put(40,30){\circle{10}}
  \put(40,30){\circle{11}}
  \put(40,30){\circle{12}}
  \put(40,30){\circle{13}}
  \put(40,30){\circle{14}}

  \put(15,10){\circle*{1}}
  \put(20,10){\circle*{2}}
  \put(25,10){\circle*{3}}
  \put(30,10){\circle*{4}}
  \put(35,10){\circle*{5}}
\end{picture}

```



La orden

$\text{\put}(x,y)\{\text{\circle}\{diámetro\}\}$
--

dibuja una circunferencia con centro  $(x, y)$  y diámetro (no radio) *diámetro*. El entorno `picture` sólo admite diámetros hasta aproximadamente 14 mm, e incluso no todos los diámetros son posibles bajo ese límite. La orden `\circle*` produce discos (círculos rellenos).

Como es el caso de segmentos de recta, uno puede recurrir a paquetes adicionales, como `eepic` o `pstricks`. Para una descripción minuciosa de estos paquetes, vea *The L<sup>A</sup>T<sub>E</sub>X Graphics Companion* [4].

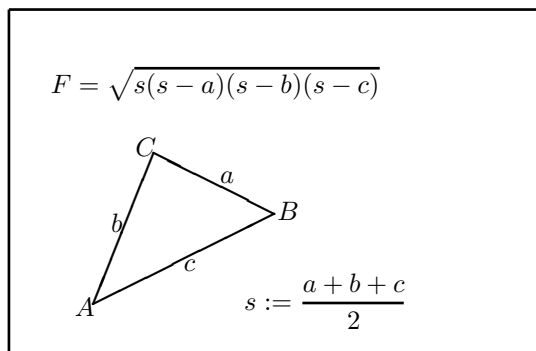
Hay también una posibilidad dentro del entorno `picture`. Si uno no tiene miedo de hacer los cálculos necesarios (o dejárselo a un programa), circunferencias y elipses arbitrarios pueden parchearse mediante curvas de Bézier. Vea *Graphics in L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub>* [17] para ejemplos y ficheros en Java.

## 5.2.5. Texto y fórmulas

```

\setlength{\unitlength}{0.8cm}
\begin{picture}(6,5)
  \thicklines
  \put(1,0.5){\line(2,1){3}}
  \put(4,2){\line(-2,1){2}}
  \put(2,3){\line(-2,-5){1}}
  \put(0.7,0.3){$A$}
  \put(4.05,1.9){$B$}
  \put(1.7,2.95){$C$}
  \put(3.1,2.5){$a$}
  \put(1.3,1.7){$b$}
  \put(2.5,1.05){$c$}
  \put(0.3,4){$F=$
    \sqrt{s(s-a)(s-b)(s-c)}$}
  \put(3.5,0.4){$\displaystyle
    s:=\frac{a+b+c}{2}$}
\end{picture}

```



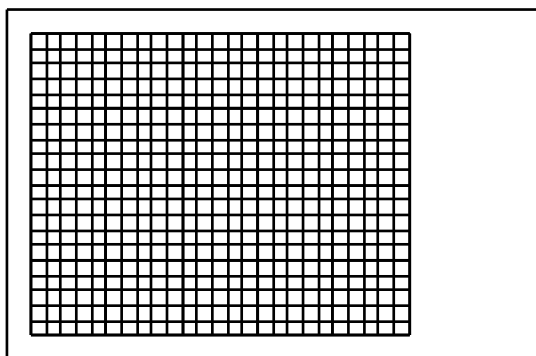
Como muestra este ejemplo, se pueden escribir texto y fórmulas en un entorno picture con la orden `\put` de la forma habitual.

5.2.6. `\multiput` y `\linethickness`

```

\setlength{\unitlength}{2mm}
\begin{picture}(30,20)
  \linethickness{0.075mm}
  \multiput(0,0)(1,0){26}%
    {\line(0,1){20}}
  \multiput(0,0)(0,1){21}%
    {\line(1,0){25}}
  \linethickness{0.15mm}
  \multiput(0,0)(5,0){6}%
    {\line(0,1){20}}
  \multiput(0,0)(0,5){5}%
    {\line(1,0){25}}
  \linethickness{0.3mm}
  \multiput(5,0)(10,0){2}%
    {\line(0,1){20}}
  \multiput(0,5)(0,10){2}%
    {\line(1,0){25}}
\end{picture}

```



La orden

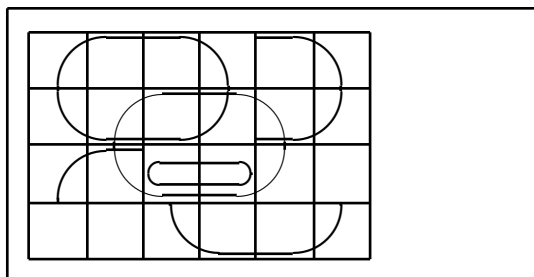
`\multiput(x,y)(\Delta x,\Delta y){n}{objeto}`

tiene 4 argumentos: el punto de inicio, el vector de traslación de un

objeto al siguiente, el número de objetos y el objeto que dibujar. La orden `\linethickness` se aplica a segmentos de recta horizontales y verticales, pero no a segmentos oblicuos ni a circunferencias. Sí se aplica, en cambio, a curvas de Bézier cuadráticas.

### 5.2.7. Óvalos

```
\setlength{\unitlength}{0.75cm}
\begin{picture}(6,4)
  \linethickness{0.075mm}
  \multiput(0,0)(1,0){7}%
    {\line(0,1){4}}
  \multiput(0,0)(0,1){5}%
    {\line(1,0){6}}
  \thicklines
  \put(2,3){\oval(3,1.8)}
  \thinlines
  \put(3,2){\oval(3,1.8)}
  \thicklines
  \put(2,1){\oval(3,1.8)[t]}
  \put(4,1){\oval(3,1.8)[b]}
  \put(4,3){\oval(3,1.8)[r]}
  \put(3,1.5){\oval(1.8,0.4)}
\end{picture}
```



La orden

```
\put(x,y){\oval(w,h)}
```

o

```
\put(x,y){\oval(w,h)[posición]}
```

produce un óvalo centrado en  $(x, y)$  y con una anchura  $w$  y altura  $h$ . Los argumentos opcionales de *posición* **t**, **b**, **l**, **r** se refieren a “top” (arriba), “bottom” (abajo), “left” (izquierda), “right”(derecha), y pueden combinarse, como ilustra el ejemplo.

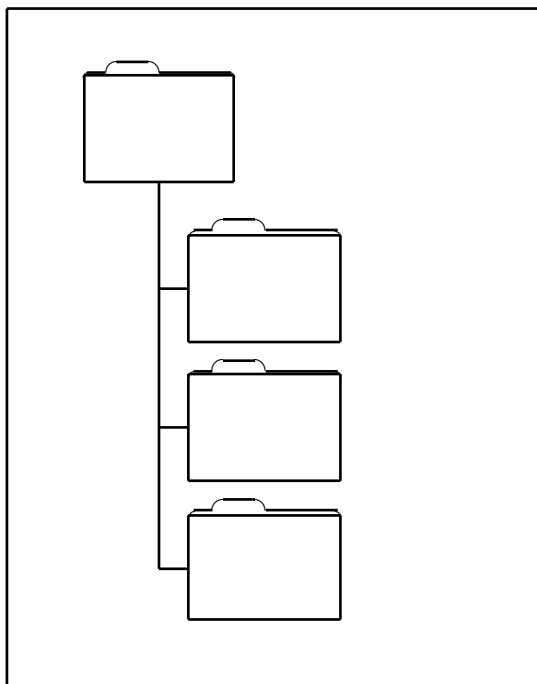
El grosor de la línea puede controlarse con dos tipos de órdenes: `\linethickness{longitud}` por un lado, `\thinlines` y `\thicklines` por el otro. Mientras `\linethickness{longitud}` se aplica sólo a líneas horizontales y verticales (y curvas de Bézier cuadráticas), `\thinlines` y `\thicklines` se aplican a segmentos de recta oblicuos y a circunferencias y óvalos.

## 5.2.8. Uso múltiple de cajas de dibujos predefinidas

```

\setlength{\unitlength}{0.5mm}
\begin{picture}(120,168)
\newsavebox{\foldera}
\savebox{\foldera}
  (40,32)[bl]{% definición
  \multiput(0,0)(0,28){2}
    {\line(1,0){40}}
  \multiput(0,0)(40,0){2}
    {\line(0,1){28}}
  \put(1,28){\oval(2,2)[tl]}
  \put(1,29){\line(1,0){5}}
  \put(9,29){\oval(6,6)[tl]}
  \put(9,32){\line(1,0){8}}
  \put(17,29){\oval(6,6)[tr]}
  \put(20,29){\line(1,0){19}}
  \put(39,28){\oval(2,2)[tr]}
}
\newsavebox{\folderb}
\savebox{\folderb}
  (40,32)[l]{% definición
  \put(0,14){\line(1,0){8}}
  \put(8,0){\usebox{\foldera}}
}
\put(34,26){\line(0,1){102}}
\put(14,128){\usebox{\foldera}}
\multiput(34,86)(0,-37){3}
  {\usebox{\folderb}}
\end{picture}

```



Una caja de dibujo puede *declararse* con la orden

```
\newsavebox{nombre}
```

y después *definirse* con

```
\savebox{nombre}(anchura,altura)[posición]{contenido}
```

y finalmente puede *dibujarse* cuantas veces se desee con

```
\put(x,y)\usebox{nombre}
```

El parámetro opcional *posición* tiene el efecto de definir el ‘punto de anclaje’ de la caja. En el ejemplo se establece a `bl`, lo que pone el punto de anclaje en la esquina inferior izquierda (bottom left) de la caja. Los otros indicadores de posición son `top` (superior) y `right` (derecha).

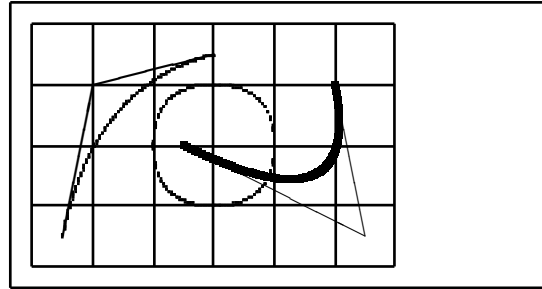


El argumento *nombre* se refiere a un espacio de almacenamiento de L<sup>A</sup>T<sub>E</sub>X y, por tanto, su aspecto ha de ser como el de una orden (lo que implica las retrobarras en el ejemplo). Las cajas de dibujo pueden anidarse: En este ejemplo, `\foldera` se usa dentro de la definición de `\folderb`.

Tiene que usarse la orden `\oval` pues la orden `\line` no funciona si la longitud del segmento es menor de 3 mm.

### 5.2.9. Curvas de Bézier cuadráticas

```
\setlength{\unitlength}{0.8cm}
\begin{picture}(6,4)
  \linethickness{0.075mm}
  \multiput(0,0)(1,0){7}
    {\line(0,1){4}}
  \multiput(0,0)(0,1){5}
    {\line(1,0){6}}
  \thicklines
  \put(0.5,0.5){\line(1,5){0.5}}
  \put(1,3){\line(4,1){2}}
  \qbezier(0.5,0.5)(1,3)(3,3.5)
  \thinlines
  \put(2.5,2){\line(2,-1){3}}
  \put(5.5,0.5){\line(-1,5){0.5}}
  \linethickness{1mm}
  \qbezier(2.5,2)(5.5,0.5)(5,3)
  \thinlines
  \qbezier(4,2)(4,3)(3,3)
  \qbezier(3,3)(2,3)(2,2)
  \qbezier(2,2)(2,1)(3,1)
  \qbezier(3,1)(4,1)(4,2)
\end{picture}
```



Como ilustra este ejemplo, dividir un círculo en 4 curvas de Bézier cuadráticas no es satisfactorio. Al menos se necesitan 8. La figura muestra de nuevo el efecto de la orden `\linethickness` en las rectas verticales u horizontales, y de las órdenes `\thinlines` y `\thicklines` en los segmentos oblicuos. También muestra que ambos tipos de órdenes afectan a las curvas de Bézier cuadráticas, de forma que cada orden se impone sobre las anteriores.

Indiquen  $P_1 = (x_1, y_1)$ ,  $P_2 = (x_2, y_2)$  los puntos extremos, y  $m_1, m_2$  las pendientes respectivas, de una curva de Bézier cuadrática. El punto de control intermedio  $S = (x, y)$  viene dado por la ecuación

$$\begin{cases} x &= \frac{m_2 x_2 - m_1 x_1 - (y_2 - y_1)}{m_2 - m_1}, \\ y &= y_i + m_i(x - x_i) \quad (i = 1, 2) \end{cases} \quad (5.1)$$

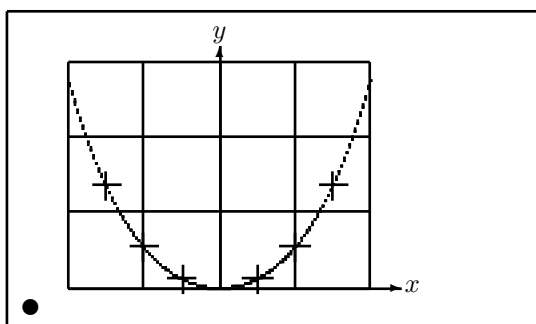
Vea *Graphics in L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub>* [17] para un programa en Java que genera la línea de órdenes `\qbezier` necesaria.

## 5.2.10. Catenaria

```

\setlength{\unitlength}{1cm}
\begin{picture}(4.3,3.6)(-2.5,-0.25)
\put(-2,0){\vector(1,0){4.4}}
\put(2.45,-.05){$x$}
\put(0,0){\vector(0,1){3.2}}
\put(0,3.35){\makebox(0,0){$y$}}
\qbezier(0.0,0.0)(1.2384,0.0)
(2.0,2.7622)
\qbezier(0.0,0.0)(-1.2384,0.0)
(-2.0,2.7622)
\linethickness{.075mm}
\multiput(-2,0)(1,0){5}
{\line(0,1){3}}
\multiput(-2,0)(0,1){4}
{\line(1,0){4}}
\linethickness{.2mm}
\put(.3,.12763){\line(1,0){.4}}
\put(.5,-.07237){\line(0,1){.4}}
\put(-.7,.12763){\line(1,0){.4}}
\put(-.5,-.07237){\line(0,1){.4}}
\put(.8,.54308){\line(1,0){.4}}
\put(1,.34308){\line(0,1){.4}}
\put(-1.2,.54308){\line(1,0){.4}}
\put(-1,.34308){\line(0,1){.4}}
\put(1.3,1.35241){\line(1,0){.4}}
\put(1.5,1.15241){\line(0,1){.4}}
\put(-1.7,1.35241){\line(1,0){.4}}
\put(-1.5,1.15241){\line(0,1){.4}}
\put(-2.5,-0.25){\circle*{0.2}}
\end{picture}

```



En esta figura, cada mitad simétrica de la catenaria  $y = \cosh x - 1$  se aproxima mediante una curva de Bézier cuadrática. La mitad derecha de la curva acaba en el punto  $(2; 27622)$ , y la pendiente allí tiene el valor  $m = 36269$ . Usando de nuevo la ecuación (5.1), podemos calcular los puntos de control intermedios. Resultan ser  $(12384; 0)$  y  $(-12384; 0)$ . Las cruces indican puntos de la catenaria *real*. El error es difícilmente perceptible, al ser menor del uno por ciento.

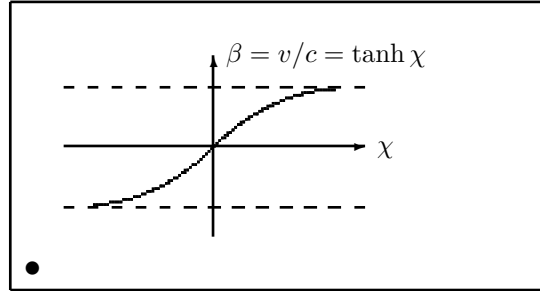
Este ejemplo incluye el uso del argumento opcional de la orden `\begin{picture}`. El dibujo se define en coordenadas “matemáticas” convenientes, mientras con la orden

```
\begin{picture}(4.3,3.6)(-2.5,-0.25)
```

a su esquina inferior izquierda (marcada con un círculo negro) se le asignan coordenadas  $(-25; -025)$ .

### 5.2.11. Rapidez en la Teoría Especial de la Relatividad

```
\setlength{\unitlength}{0.8cm}
\begin{picture}(6,4)(-3,-2)
  \put(-2.5,0){\vector(1,0){5}}
  \put(2.7,-0.1){\chi}
  \put(0,-1.5){\vector(0,1){3}}
  \multiput(-2.5,1)(0.4,0){13}
    {\line(1,0){0.2}}
  \multiput(-2.5,-1)(0.4,0){13}
    {\line(1,0){0.2}}
  \put(0.2,1.4)
    {\beta=v/c=\tanh\chi}
  \qbezier(0,0)(0.8853,0.8853)
    (2,0.9640)
  \qbezier(0,0)(-0.8853,-0.8853)
    (-2,-0.9640)
  \put(-3,-2){\circle*{0.2}}
\end{picture}
```



Los puntos de control de las dos curvas de Bézier se calcularon con las fórmulas (5.1). La rama positiva se determina con  $P_1 = (0; 0)$ ,  $m_1 = 1$  y  $P_2 = (2; \tanh 2)$ ,  $m_2 = 1/\cosh^2 2$ . De nuevo, el dibujo se define en coordenadas matemáticas convenientes, y a la esquina inferior izquierda se le asignan las coordenadas matemáticas  $(-3; -2)$  (círculo negro).

## 5.3. Los paquetes graficos PGF y TikZ

Hoy en día todos los sistemas de generación de salida de  $\text{\LaTeX}$  pueden crear agradables gráficos vectoriales, es sólo que las interfaces son bastante diversas. El paquete `pgf` proporciona una capa de abstracción sobre estas interfaces. El paquete `pgf` viene con un gran manual/tutorial propio [18]. Así que sólo vamos a arañar la superficie del paquete con esta pequeña sección.

El paquete `pgf` viene con un lenguaje de alto nivel de acceso proporcionado por el paquete `tikz`. `TikZ` proporciona comandos altamente eficientes para dibujar gráficos correctamente dentro de su documento. Utilice el entorno `tikzpicture` para agrupar sus comandos `TikZ`.

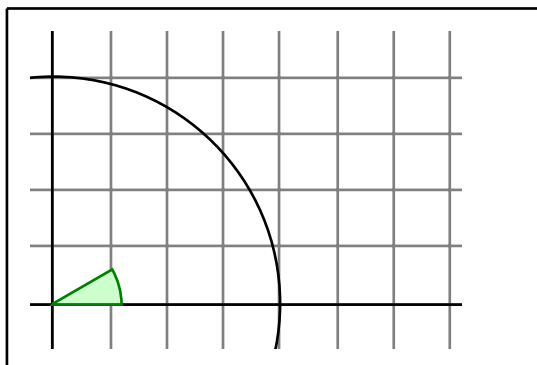
Como se mencionó anteriormente, hay un excelente manual para `pgf` y compañía. Así que en lugar de explicar realmente cómo funciona, les mostraré algunos ejemplos para que puedan obtener una primera impresión de cómo funciona esta herramienta.

En primer lugar un diagrama simple sin sentido.

```

\begin{tikzpicture}[scale=3]
  \clip (-0.1,-0.2)
    rectangle (1.8,1.2);
  \draw[step=.25cm,gray,very thin]
    (-1.4,-1.4) grid (3.4,3.4);
  \draw (-1.5,0) -- (2.5,0);
  \draw (0,-1.5) -- (0,1.5);
  \draw (0,0) circle (1cm);
  \filldraw[fill=green!20!white,
    draw=green!50!black]
    (0,0) -- (3mm,0mm)
      arc (0:30:3mm) -- cycle;
\end{tikzpicture}

```



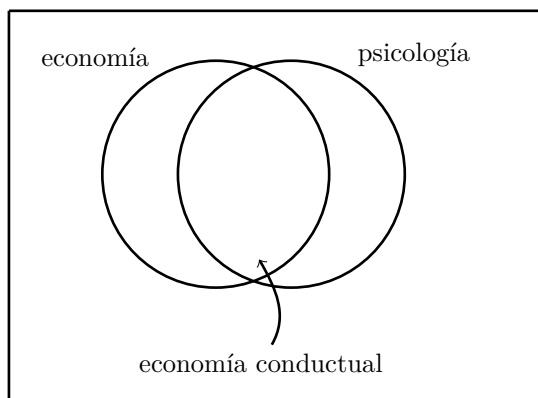
Observe el carácter punto y coma (;). Separa los comandos individuales.

Un simple diagrama de Venn

```

\shorthandoff{:}
\begin{tikzpicture}
  \node[circle,draw,
    minimum size=3cm,
    label=120:{economía}]
    at (0,0) {};
  \node[circle,draw,
    minimum size=3cm,
    label=60:{psicología}]
    at (1,0) {};
  \node (i) at (0.5,-1) {};
  \node at (0.6,-2.5)
    {economía conductual}
    edge[->,thick,
      out=60,in=-60] (i);
\end{tikzpicture}

```



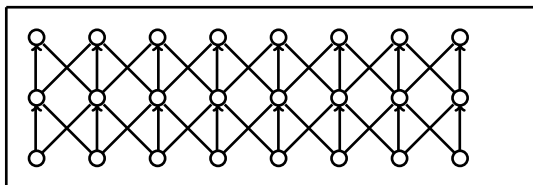
Si está utilizando TikZ en relación con **babel** algunos de los caracteres utilizados en el lenguaje TikZ pueden ser modificados por **babel**, lo que lleva a errores singulares. Para contrarrestar este problema, agregue el comando `\shorthandoff` a su código.

Observe los bucles `foreach` en el siguiente ejemplo.

```

\begin{tikzpicture}[scale=0.8]
  \tikzstyle{v}=[circle, minimum size=2mm,inner sep=0pt,draw]
  \foreach \i in {1,...,8}
    \foreach \j in {1,...,3}
      \node[v]
        (G-\i-\j) at (\i,\j) {};
  \foreach \i in {1,...,8}
    \foreach \j/\o in {1/2,2/3}
      \draw[->]
        (G-\i-\j) -- (G-\i-\o);
  \foreach \i/\n in
    {1/2,2/3,3/4,4/5,5/6,6/7,7/8}
    \foreach \j/\o in {1/2,2/3} {
      \draw[->] (G-\i-\j) -- (G-\n-\o);
      \draw[->] (G-\n-\j) -- (G-\i-\o);
    }
}
\end{tikzpicture}

```

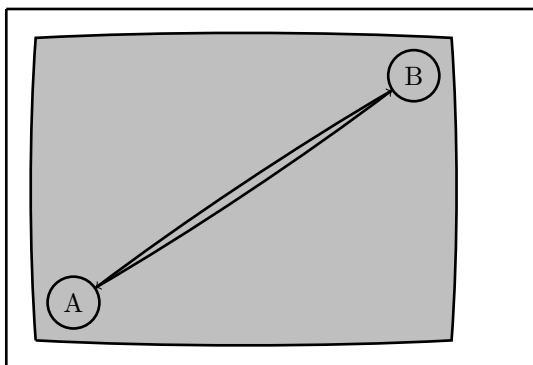


Con la orden `\usetikzlibrary` en el preámbulo puede habilitar una amplia variedad de características adicionales para dibujar formas especiales, como esta caja que está ligeramente curvada.

```

\usetikzlibrary{%
  decorations.pathmorphing}
\begin{tikzpicture}[
  decoration={bent,aspect=.3}]
  \draw [decorate,fill=lightgray]
    (0,0) rectangle (5.5,4);
  \node[circle,draw]
    (A) at (.5,.5) {A};
  \node[circle,draw]
    (B) at (5,3.5) {B};
  \draw[->,decorate] (A) -- (B);
  \draw[->,decorate] (B) -- (A);
\end{tikzpicture}

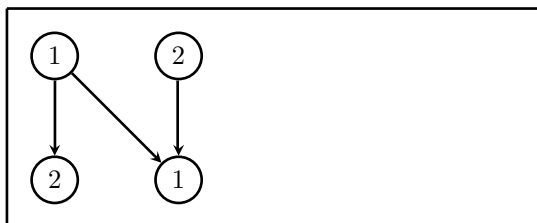
```



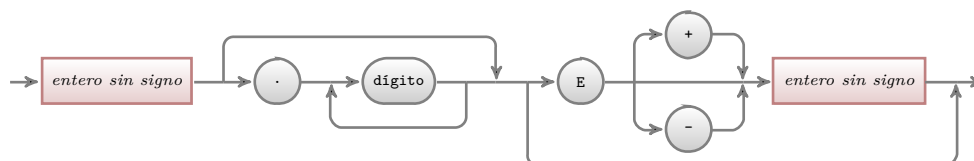
```

\usetikzlibrary{positioning}
\begin{tikzpicture}[xscale=6,
  yscale=8,>=stealth]
  \tikzstyle{v}=[circle,
    minimum size=1mm,draw,thick]
  \node[v] (a) {$1$};
  \node[v] (b) [right=of a] {$2$};
  \node[v] (c) [below=of a] {$2$};
  \node[v] (d) [below=of b] {$1$};
  \draw[thick,->]
    (a) to node {} (c);
  \draw[thick,->]
    (a) to node {} (d);
  \draw[thick,->]
    (b) to node {} (d);
\end{tikzpicture}

```



Incluso puede dibujar diagramas de sintaxis que se ven como si vinieran directamente de un libro sobre programación en Pascal. El código es un poco más intimidante que el ejemplo anterior, por lo que sólo mostraré el resultado. Si usted echa un vistazo en la documentación de `pgf` encontrará un tutorial detallado sobre la elaboración de este diagrama exacto.



Y hay más, si tiene que dibujar gráficas de datos o funciones numéricas, usted puede echar un vistazo más de cerca al paquete `pgfplot`. Ofrece todo lo necesario para dibujar gráficos. Incluso puede llamar al comando externo `gnuplot` para evaluar las funciones reales que trazó en el gráfico.

Para una mayor inspiración, asegúrese de visitar la excelente página <http://www.texample.net/tikz/> de Kjell Magne Fauske que contiene un depósito cada vez más grande de hermosos gráficos y otro tipo de código  $\text{\LaTeX}$ . En  $\text{\TeX}$ ample.net también encontrará una [lista de herramientas](#) para trabajar con PGF/TikZ de modo que usted no tenga que escribir todo ese código a mano.



## Capítulo 6

# Personalización de L<sup>A</sup>T<sub>E</sub>X

Los documentos producidos mediante las órdenes que ha aprendido hasta este punto parecerán aceptables a una amplia audiencia. Aunque no tienen un aspecto extraordinario, obedecen todas las reglas establecidas de composición correcta, lo que los hará fáciles de leer y plácidos a la vista.

Sin embargo, hay situaciones donde L<sup>A</sup>T<sub>E</sub>X no proporciona una orden o entorno que cubra sus necesidades, o la salida producida por algunas órdenes existentes puede no satisfacer sus expectativas.

En este capítulo, se darán algunas pistas para enseñar a L<sup>A</sup>T<sub>E</sub>X nuevos trucos y hacerle producir salidas con diferente aspecto del producido por omisión.

### 6.1. Nuevas órdenes, entornos y paquetes

Puede haber notado que todas las órdenes que presento en este libro se componen en una caja, y que se muestran en el índice al final del libro. En lugar de usar directamente las órdenes L<sup>A</sup>T<sub>E</sub>X necesarias para conseguirlo, he creado un paquete en que defino nuevas órdenes y entornos con este propósito. Ahora puedo escribir simplemente:

```
\begin{lscommand}  
\ci{dum}  
\end{lscommand}
```



\dum

En este ejemplo, estoy usando tanto un nuevo entorno llamado `lscommand`, que es responsable de dibujar la caja alrededor de la orden, y una nueva orden llamada `\ci`, que compone el nombre de la orden y hace la correspondiente entrada en el índice. Puede comprobarlo buscando la orden `\dum` en el índice al final del libro, donde pude encontrar una entrada para `\dum`, apuntando a cada página donde he mencionado la orden `\dum`.

Si alguna vez decido que no me gusta que las órdenes se compongan en una caja, puedo simplemente cambiar la definición del entorno `lscommand`



para crear un nuevo aspecto. Esto es mucho más fácil que ir por todo el documento localizando todos los lugares en que he usado comandos L<sup>A</sup>T<sub>E</sub>X genéricos para dibujar una caja alrededor de una palabra.

### 6.1.1. Órdenes nuevas

Para añadir sus órdenes nuevas, use la orden

`\newcommand{nombre}[núm]{definición}`

Básicamente, lo orden requiere dos argumentos: el *nombre* de la orden que quiere crear, y la *definición* de la orden. El argumento *núm* entre corchetes es opcional e indica el número de argumentos que toma la nueva orden (hasta 9 son posibles). Si no se indica el valor es 0, es decir, no se permiten argumentos.

Los siguientes dos ejemplos deberían ayudarle a entender la idea. El primer ejemplo define una nueva orden llamada `\intc`. Es la abreviatura de “La introducción no-tan-corta a L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub>”. Tal orden podría ser útil si tuviera que escribir el título del libro una y otra vez.

```
\newcommand{\intc}{La
  introducción no-tan-corta a
  \LaTeXe}
Esto es ‘‘\intc’’ \ldots{}
‘‘\intc’’
```

Esto es “La introducción no-tan-corta a L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub>” ... “La introducción no-tan-corta a L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub>”

El siguiente ejemplo ilustra cómo definir una orden nueva que toma un argumento. Los caracteres `#1` se sustituyen por el argumento indicado. Si quisiera usar un segundo argumento, use `#2` y así sucesivamente.

```
\newcommand{\txsit}[1]
{Esta es la Introducción
  \emph{#1}-corta a \LaTeXe}
% en el cuerpo del documento:
\begin{itemize}
  \item \txsit{no-tan}
  \item \txsit{súper}
\end{itemize}
```

- Esta es la Introducción *no-tan*-corta a L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub>
  - Esta es la Introducción *súper*-corta a L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub>

L<sup>A</sup>T<sub>E</sub>X no le permitirá crear una nueva orden sobre una ya existente. Pero hay una orden especial en el caso de que explícitamente quisiera reemplazarla: `\renewcommand`. Usa la misma sintaxis que la orden `\newcommand`.

En ciertos casos puede querer usar la orden `\providecommand`. Funciona como `\newcommand` y hace que la orden sea definida si aún no existe, pero no hace nada si ya estaba definida.

Hay algunos puntos que comentar sobre los espacios que siguen a las órdenes de L<sup>A</sup>T<sub>E</sub>X. Vea la página 5 para más información.

### 6.1.2. Nuevos entornos

Similar a la orden `\newcommand`, hay una orden para crear sus propios entornos. La orden `\newenvironment` usa la siguiente sintaxis:

$$\text{\newenvironment\{nombre\}[n\acute{u}m]\{antes\}\{despu\acute{e}s\}}$$

También `\newenvironment` puede tener un argumento opcional. El material indicado en el argumento *antes* se procesa antes de que se procese el texto del entorno. El material en el argumento *después* se procesa cuando se encuentra la orden `\end\{nombre\}`.

El ejemplo siguiente ilustra el uso de la orden `\newenvironment`.

```
\newenvironment{king}
{\rule{1ex}{1ex}%
 \hspace{\stretch{1}}}
{\hspace{\stretch{1}}%
 \rule{1ex}{1ex}}
```

■ Mis humildes ideas... ■

```
\begin{king}
Mis humildes ideas\ldots
\end{king}
```

El argumento *núm* se usa igual que con la orden `\newcommand`.  $\text{\LaTeX}$  se asegura de que usted no defina un entorno que ya existe; pero si quiere alguna vez cambiar un entorno existente, puede usar la orden `\renewenvironment`. Usa la misma sintaxis que la orden `\newenvironment`.

La orden usada en este ejemplo se explicará más tarde. Para la orden `\rule` véase la página 129, para `\stretch` vaya a la página 122, y puede hallar más información sobre `\hspace` en la página 122.

### 6.1.3. Espacio extra

Al crear un entorno nuevo puede hallar dificultades en el manejo del espacio adicional, que puede llegar a tener efectos fatales. Por ejemplo, cuando quiera crear un entorno para títulos que suprima su propia sangría así como la del siguiente párrafo. La orden `\ignorespaces` en el bloque de comienzo del entorno hará que éste prescinda de cualquier espacio tras ejecutar el bloque de comienzo. El bloque final requiere un poco más de cuidado porque tiene lugar un proceso especial al final del entorno. La orden `\ignorespacesafterend` hará que  $\text{\LaTeX}$  ejecute `\ignorespaces` después de que el proceso especial tenga lugar.

```
\newenvironment{simple}%
{\noindent}%
{\par\noindent}

\begin{simple}
Mire el espacio\\a la izquierda.
\end{simple}
También\\aquí.
```

Mire el espacio  
a la izquierda.

También  
aquí.

```
\newenvironment{correct}%
{\noindent\ignorespaces}%
{\par\noindent%
\ignorespacesafterend}

\begin{correct}
Sin espacio\\a la izquierda.
\end{correct}
También\\aquí.
```

Sin espacio  
a la izquierda.

También  
aquí.

#### 6.1.4. Línea de órdenes L<sup>A</sup>T<sub>E</sub>X

Si trabaja en un sistema operativo estilo POSIX (GNU o UNIX), quizás use `\Makefile` para compilar sus documentos de L<sup>A</sup>T<sub>E</sub>X. Entonces podría ser interesante producir diferentes versiones del mismo documento llamando a L<sup>A</sup>T<sub>E</sub>X con diversos parámetros en la línea de órdenes. Si añade la siguiente estructura a su documento:

```
\usepackage{ifthen}
\ifthenelse{\equal{\blancoynegro}{verdadero}}{
% modo "blanco y negro"; hacer algo..
}{
% modo "color"; hacer algo diferente..
}
```

Ahora puede llamar a L<sup>A</sup>T<sub>E</sub>X así:

```
latex '\newcommand{\blancoynegro}{verdadero}\input{test.tex}'
```

Primero se define la orden `\blancoynegro` y después se lee el fichero real. Poniendo `\blancoynegro` a falso se producirá la versión en color del documento.

#### 6.1.5. Su propio paquete

Si define muchos nuevos entornos y órdenes, el preámbulo de su documento se hará muy largo. En situaciones así es buena idea crear un paquete L<sup>A</sup>T<sub>E</sub>X

que contenga todas sus definiciones de órdenes y entornos. Puede usar después la orden `\usepackage` para cargar el paquete en su documento actual o en otros similares.

---

```
% Paquete Demo de Tobias Oetiker
\ProvidesPackage{demopack}
\newcommand{\intc}{La introducción no-tan-corta
a \LaTeXe}
\newcommand{\txsit}[1]{La introducción \emph{#1}-corta
a \LaTeXe}
\newenvironment{king}{\begin{quote}}{\end{quote}}
```

---

Figura 6.1: Paquete de ejemplo.

Escribir un paquete básicamente consiste en copiar el contenido del preámbulo de su documento en un fichero separado con un nombre que termine en `.sty`. Hay una orden especial,

`\ProvidesPackage{nombre paquete}`

para usar justo al principio de su fichero de paquete. `\ProvidesPackage` dice a  $\text{\LaTeX}$  el nombre del paquete y le permite emitir un mensaje de error notable cuando intente incluir el paquete dos veces. La figura 6.1 muestra un pequeño paquete de ejemplo que contiene órdenes definidas en ejemplos anteriores.

## 6.2. Fundiciones y tamaños

### 6.2.1. Órdenes que cambian la fundición

$\text{\LaTeX}$  escoge la fundición y el tamaño de fundición apropiados basándose en la estructura lógica del documento (secciones, notas al pie, ...). En algunos casos, quizá desee cambiar fundiciones y tamaños a mano. Para hacerlo, puede usar las órdenes listadas en los cuadros 6.1 y 6.2. El tamaño real de cada fundición es una cuestión de diseño y depende de la clase de documento y de sus opciones. El cuadro 6.3 muestra los tamaños absolutos en puntos para estas órdenes según se implementan en las clases de documentos normales.

```
{\small Pequeña \textbf{negrita}
del África tropical,}
{\Large grande y \textit{cursi}va
eres tú ya.}
```

Pequeña **negrita** del África tropical,  
grande y *cursiva* eres tú ya.

Una característica importante de L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> es que los atributos de fundición son independientes. Esto significa que puede poner órdenes para cambiar el tamaño o incluso la fundición, y todavía se mantendrán los atributos de negrita o cursiva establecidos anteriormente.

En *modo mates* puede usar las *órdenes* de cambio de fundición para salir temporalmente del *modo mates* e introducir texto normal. Si quiere cambiar a otra fundición para composición de mates necesita otro conjunto especial de órdenes; véase el cuadro 6.4.

En relación a las órdenes de tamaño de fundición, las llaves representan un papel significativo. Se usan para construir *grupos*. Los grupos limitan el alcance de la mayoría de las órdenes de L<sup>A</sup>T<sub>E</sub>X.

Adora los {\LARGE grandes y  
\small pequeños} placeres}.

Adora los grandes y pequeños pla-  
ceres.

Las órdenes de tamaño de fundición también cambian el espaciado entre renglones, pero sólo si el párrafo termina dentro del ámbito de la orden de tamaño de fundición. La llave de cierre } debería por tanto no llegar demasiado pronto. Fíjese en la posición de la orden \par en los siguientes

Cuadro 6.1: Fundiciones.

\textrm{...}	rematada	\textsf{...}	palo seco
\texttt{...}	de máquina		
\textmd{...}	peso medio	\textbf{...}	<b>negrita</b>
\textup{...}	recta	\textit{...}	<i>cursiva</i>
\textsl{...}	oblicua	\textsc{...}	VERSALITAS
\emph{...}	<i>destacada</i>	\textnormal{...}	por omisión

Cuadro 6.2: Tamaños de fundición.

\tiny	fundición minúscula	\Large	más grande
\scriptsize	fundición muy pequeña	\LARGE	muy grande
\footnotesize	bastante pequeña	\huge	enorme
\small	fundición pequeña		
\normalsize	fundición normal	\Huge	la más
\large	fundición grande		

Cuadro 6.3: Tamaños absolutos en puntos para las clases normales.

tamaño	10pt (por omisión)	opción 11pt	opción 12pt
<code>\tiny</code>	5pt	6pt	6pt
<code>\scriptsize</code>	7pt	8pt	8pt
<code>\footnotesize</code>	8pt	9pt	10pt
<code>\small</code>	9pt	10pt	11pt
<code>\normalsize</code>	10pt	11pt	12pt
<code>\large</code>	12pt	12pt	14pt
<code>\Large</code>	14pt	14pt	17pt
<code>\LARGE</code>	17pt	17pt	20pt
<code>\huge</code>	20pt	20pt	25pt
<code>\Huge</code>	25pt	25pt	25pt

Cuadro 6.4: Fundiciones para mates.

<code>\mathrm{...}</code>	Fundición Rematada
<code>\mathbf{...}</code>	<b>Fundición Negrita</b>
<code>\mathsf{...}</code>	Fundición Palo Seco
<code>\mathtt{...}</code>	Fundición De Máquina
<code>\mathit{...}</code>	<i>Fundición Cursiva</i>
<code>\mathcal{...}</code>	<i>FUNDICIÓN CALIGRÁFICA</i>
<code>\mathnormal{...}</code>	<i>Fundición Normal</i>

dos ejemplos.<sup>1</sup>

```
{\Large ¡No lea esto!  
No es verdad.  
¡Puede creerme!}\par}
```

¡No lea esto! No es verdad.  
¡Puede creerme!

```
{\Large Tampoco esto es verdad.  
Mas recuerde qué mendaz soy.}\par}
```

Tampoco esto es verdad. Mas  
recuerde qué mendaz soy.

Si quiere activar una orden de cambio de tamaño para un párrafo entero de texto o incluso más, puede usar la sintaxis de entorno para las órdenes de cambio de fundición.

```
\begin{Large}  
Esto no es verdad, pero  
qué diantres cabe esperar  
en estos tiempos...\par  
\end{Large}
```

Esto no es verdad, pero qué  
diantres cabe esperar en estos  
tiempos...

Esto le ahorrará andar contando llaves.

### 6.2.2. ¡Atención: peligro!

Como se comenta al principio de este capítulo, es peligroso sembrar el documento con órdenes explícitas como éstas, pues funcionan contra la idea básica de L<sup>A</sup>T<sub>E</sub>X, que es separar la estructura de su documento del aspecto visual. Esto significa que si usted usa la misma orden de cambio de fundición en varios lugares para componer un tipo especial de información, debería usar `\newcommand` para definir una “orden lógica encubridora” para la orden de cambio de fundición.

```
\newcommand{\ojo}[1]{%  
  \textbf{#1}}  
No \ojo{entre} en esta sala; está  
ocupada por \ojo{máquinas} de  
origen y propósito desconocidos.
```

No **entre** en esta sala; está ocupada por  
**máquinas** de origen y propósito desconocidos.

Este enfoque tiene la ventaja de que usted puede decidir en una etapa posterior que quiere usar alguna representación visual de peligro distinta de `\textbf`, sin tener que recorrer todo el documento identificando cada aparición de `\textbf` y después deduciendo si ahí se usó para señalar un peligro o por alguna otra razón.

---

<sup>1</sup>`\par` equivale a un renglón en blanco.

### 6.2.3. Consejo

Para concluir este viaje al mundo de las fundiciones y sus tamaños, acepte este humilde consejo:

**¡Recuerde!** *Cuantas M<sup>Á</sup>S fundiciones use en un documento, tanto más LEGIBLE y lindo será.*

## 6.3. Espaciado

### 6.3.1. Espacio entre renglones

Si quiere usar mayor espacio entre renglones, puede cambiar su valor poniendo la orden

```
\linespread{factor}
```

en el preámbulo de su documento. Use `\linespread{1.3}` para espaciado de “uno y medio” y `\linespread{1.6}` para espaciado “doble”. Normalmente los renglones no se separan, así que el factor por omisión es 1.

Tenga en cuenta que el efecto de la orden `\linespread` es bastante drástico y no apropiado para publicar un trabajo. Así que si tiene una buena razón para cambiar el espacio entre renglones quizá prefiera usar la orden:

```
\setlength{\baselineskip}{1.5\baselineskip}
```

```
{\setlength{\baselineskip}%
{1.5\baselineskip}
Este párrafo está compuesto con
el salto de línea base puesto a
1,5 de lo que era antes. Fíjese
en la orden par al final del
párrafo.\par}
```

Este párrafo tiene un propósito claro: mostrar que, una vez se cierran las llaves, todo vuelve a la normalidad.

Este párrafo está compuesto con el salto de línea base puesto a 1,5 de lo que era antes. Fíjese en la orden par al final del párrafo.

Este párrafo tiene un propósito claro: mostrar que, una vez se cierran las llaves, todo vuelve a la normalidad.

### 6.3.2. Formato de párrafo

En  $\text{\LaTeX}$ , hay dos parámetros que influyen en el aspecto del párrafo. Poniendo una definición



```
\setlength{\parindent}{0pt}
\setlength{\parskip}{1ex plus 0.5ex minus 0.2ex}
```

en el preámbulo del fichero de entrada, puede cambiar el aspecto de los párrafos. Estas dos órdenes incrementan el espacio entre dos párrafos y establecen la sangría de párrafo a cero.

Las partes `plus` y `minus` de la longitud de arriba dicen a T<sub>E</sub>X que puede comprimir y expandir el salto entre párrafos la cantidad indicada, si es necesario para ajustar apropiadamente los párrafos en la página.

En algunos países europeos los párrafos suelen separarse algo y no se sangran. Pero tenga en cuenta que esto tiene su efecto en el índice general; sus renglones se espaciarán más en ese caso. Para evitarlo, puede mover las dos órdenes del preámbulo a un lugar en su documento detrás de la orden `\tableofcontents` o no usarlo en absoluto, porque verá que muchos libros profesionales usan sangría y no espacio para separar párrafos.

Si quiere sangrar un párrafo que no está sangrado, puede usar

`\indent`

al principio del párrafo. Obviamente, sólo tendrá efecto cuando `\parindent` no valga cero. Para sangrar el primer párrafo tras cada título de sección, use el paquete `indentfirst` del lote ‘tools’.

Para crear un párrafo no sangrado, puede usar

`\noindent`

como primera orden del párrafo. Puede ser útil si empieza un documento con texto de párrafo y no con una orden de sección.

### 6.3.3. Espacio horizontal

L<sup>A</sup>T<sub>E</sub>X determina los espacios entre palabras y automáticamente. Para añadir espacio horizontal, use:







`\hspace{longitud}`

Si dicho espacio debiera mantenerse incluso si cae al final o al principio de renglón, use `\hspace*` en lugar de `\hspace`. La *longitud* en el caso más simple es sólo un número más una unidad. Las unidades más importantes se listan en el cuadro 6.5.

Éste `\hspace{1.5cm}` es un espacio  
de 1,5 cm.

Éste                      es un espacio de 1,5 cm.

Cuadro 6.5: Unidades T<sub>F</sub>X.

mm	milímetro $\approx 1/25$ pulgada	
cm	centímetro = 10 mm	
in	pulgada = 25,4 mm	
pt	punto $\approx 1/72$ pulgada $\approx \frac{1}{3}$ mm	
em	$\approx$ anchura de una 'M' en la fundición actual	
ex	$\approx$ altura de una 'x' en la fundición actual	

La orden

---

 $\stretch{n}$ 

genera espacio especial, que se expande hasta llenar todo el espacio sobrante en un renglón. Si dos órdenes `\hspace{\stretch{n}}` tienen lugar en el mismo renglón, los espacios crecen proporcionalmente a sus argumentos.

X X X X

Al usar espacio horizontal junto con texto, puede tener sentido hacer que el espacio ajuste su tamaño en relación con el tamaño de la fundición actual. Esto puede hacerse usando las unidades relativas a la fundición `em` y `ex`:

gran y  
pequeña y

#### 6.3.4. Espacio vertical

L<sup>A</sup>T<sub>E</sub>X determina automáticamente el espacio entre párrafos, secciones, subsecciones, etc. Si es necesario, puede añadirse espacio vertical adicional *entre dos párrafos* con la orden:

---

\vspace{longitud}

Esta orden debería usarse normalmente entre dos renglones vacíos. Si el espacio debe preservarse en lo alto o en lo bajo de la página, use la versión de la orden con asterisco, `\vspace*`, en lugar de `\vspace`.

La orden `\stretch`, acompañada de `\pagebreak`, puede usarse para escribir texto en el último renglón de una página, o para centrar texto verticalmente en una página.

Algo de texto...

`\vspace{\stretch{1}}`

Esto va en la última línea de la página. `\pagebreak`

Espacio adicional entre dos líneas del *mismo* párrafo o dentro de una tabla se indica con la orden

`\[longitud]`

Con `\bigskip` y `\smallskip` puede saltar una cantidad predefinida de espacio vertical sin tener que preocuparse de números exactos.

## 6.4. Composición de la página

L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> le permite indicar el tamaño del papel en la orden `\documentclass`. Después calcula los márgenes adecuados, pero a veces usted no estará contento con los valores predefinidos. Naturalmente, puede cambiarlos.

La figura 6.2 muestra todos los parámetros que pueden cambiarse. La figura se creó con el paquete `layout` del lote ‘tools’.<sup>2</sup>

**¡ESPERE!** Antes de lanzarse al frenesí de “Hagamos esa página estrecha un poco más ancha”, dedique unos segundos a pensar. Como muchas cosas en L<sup>A</sup>T<sub>E</sub>X, hay una buena razón para que el aspecto de la página sea como es.

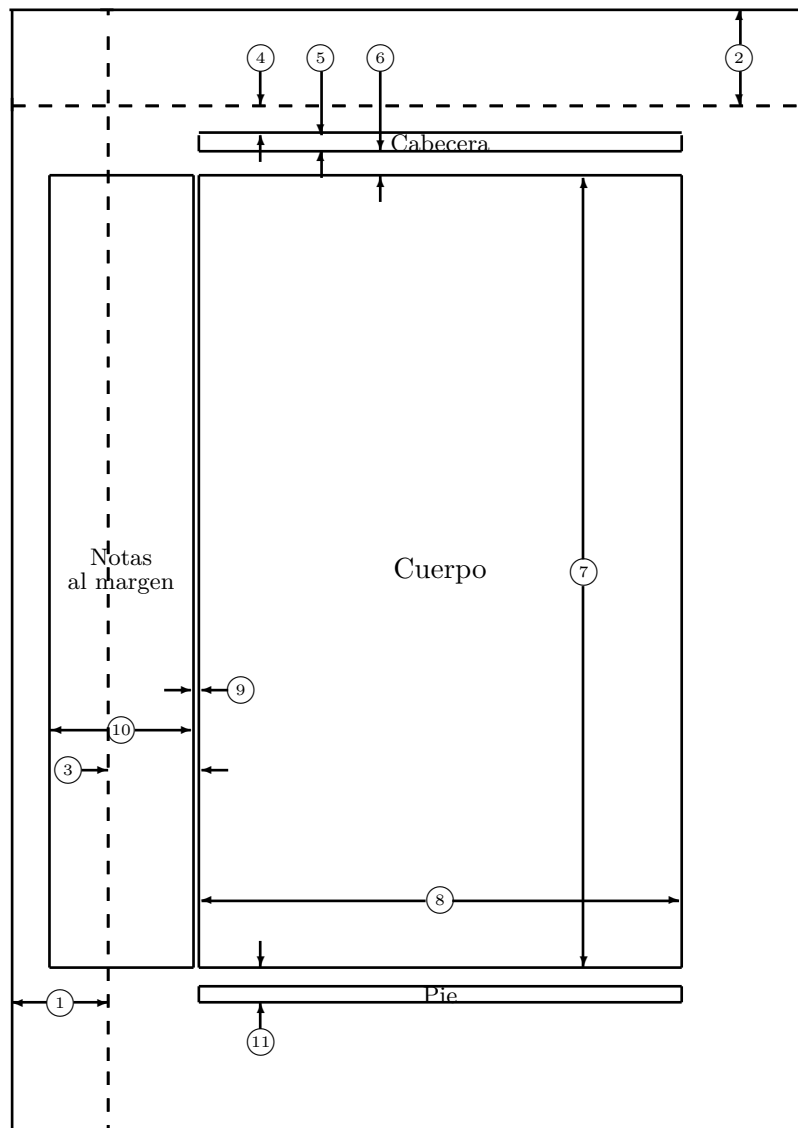
Por supuesto, comparada con su página recién salida de un paquete ofimático (como OpenOffice Writer o MS Word), parece horrorosamente estrecha. Pero eche un vistazo a su libro favorito<sup>3</sup> y cuente el número de caracteres en una línea de texto normal. Hallará que no hay más de en torno a 66 caracteres en cada renglón. Ahora haga lo mismo con su página de L<sup>A</sup>T<sub>E</sub>X; verá lo mismo. La experiencia muestra que la lectura se vuelve difícil en cuanto hay más caracteres por renglón. Es así porque a los ojos les resulta difícil moverse desde el final de un renglón al principio del siguiente. Es la misma razón por la que los periódicos se componen en múltiples columnas. Así que si incrementa la anchura de su texto, tenga en cuenta que está haciendo la vida más difícil a los lectores de su documento.

Si de cualquier forma quiere hacerlo, L<sup>A</sup>T<sub>E</sub>X proporciona dos órdenes para cambiar estos parámetros. Se usan normalmente en el preámbulo del documento.

---

<sup>2</sup>`macros/latex/required/tools`

<sup>3</sup>Me refiero a un libro real impreso y producido por una editorial con reputación.



- |  |  |
|--|--|
| 1 una pulgada + \hoffset                                     | 2 una pulgada + \voffset   |
| 3 \oddsidemargin = 22pt<br>o \evensidemargin                 | 4 \topmargin = 22pt  |
| 5 \headheight = 12pt   | 6 \headsep = 19pt  |
| 7 \textheight = 595pt  | 8 \textwidth = 360pt   |
| 9 \marginparsep = 7pt  | 10 \marginparwidth = 106pt<br>\marginparpush = 5pt (no se muestra)<br>\voffset = 0pt |
| 11 \footskip = 27pt<br>\hoffset = 0pt<br>\paperwidth = 597pt | \paperheight = 845pt   |

Figura 6.2: Parámetros de composición de la página.

La primera orden asigna un valor fijo a cualquiera de los parámetros:

`\setlength{parámetro}{longitud}`

La segunda orden añade longitud a cualquier parámetro:

`\addtolength{parámetro}{longitud}`

Esta segunda orden es de hecho más útil que la orden `\setlength`, pues puede usted así trabajar en relación a los valores establecidos. Para añadir un centímetro a la anchura total del texto, pongo las siguientes órdenes en el preámbulo del documento:

```
\addtolength{\hoffset}{-0.5cm}
\addtolength{\textwidth}{1cm}
```

En este contexto, quizá quiera mirar el paquete `calc`. Le permite usar operaciones aritméticas en el argumento de `\setlength` y en otros lugares donde puede introducir valores numéricos en argumentos de funciones.

## 6.5. Más diversión con las longitudes

Siempre que sea posible, evite usar longitudes absolutas en los documentos L<sup>A</sup>T<sub>E</sub>X. Intente basar las cosas en la anchura o altura de otros elementos de la página. Para la anchura de una figura puede referirse a `\textwidth` al componer la página.

Las siguientes 3 órdenes le permiten determinar la anchura, altura y profundidad de una cadena de texto.

`\settoheight{variable}{texto}`  
`\settodepth{variable}{texto}`  
`\settowidth{variable}{texto}`

El ejemplo siguiente muestra una posible aplicación de estas órdenes.

```

\flushleft
\newenvironment{vardesc}[1]{%
  \settowidth{\parindent}{#1:\ }
  \makebox[Opt][r]{#1:\ }}{}

\begin{displaymath}
a^2+b^2=c^2
\end{displaymath}

\begin{vardesc}{Donde}$a$,
$b$ -- son adyacentes al ángulo
recto de un triángulo rectángulo.

$c$ -- es la hipotenusa del
triángulo, y

$d$ -- no sale aquí
en absoluto.
\end{vardesc}

```

$$a^2 + b^2 = c^2$$

Donde:  $a$ ,  $b$  – son adyacentes al ángulo recto de un triángulo rectángulo.

$c$  – es la hipotenusa del triángulo,

y

$d$  – no sale aquí en absoluto.

## 6.6. Cajas

$\text{\LaTeX}$  construye sus páginas colocando cajas. En principio, cada letra es una cajita, que se pega a otras letras para formar palabras. Éstas se pegan de nuevo a otras palabras, pero con un pegamento especial, que es tan elástico que una serie de palabras puede comprimirse o expandirse para rellenar exactamente un renglón de la página.

Esto es una simplificación de lo que realmente ocurre, pero realmente ocurre:  $\text{\TeX}$  trabaja con pegamento y cajas. Las letras no son las únicas cosas que son cajas. Puede poner virtualmente cualquier cosa en una caja, incluso otras cajas. Cada caja será manejada por  $\text{\LaTeX}$  como si fuera una simple letra.

En los capítulos anteriores ya ha encontrado algunas cajas, aunque no lo parezcan. Los entornos `tabular` e `\includegraphics`, por ejemplo, producen cajas. Esto significa que puede usted fácilmente colocar dos tablas o imágenes una al lado de la otra. Basta con asegurarse de que su anchura combinada no excede la anchura del texto.

Puede también empaquetar un párrafo de su elección en una caja con la

orden

```
\parbox[pos]{anchura}{texto}
```

o el entorno

```
\begin{minipage}[pos]{anchura} texto \end{minipage}
```

El parámetro `pos` puede tomar una de las letras `c`, `t` o `b` para controlar la alineación vertical de la caja, relativa a la línea base del texto que la rodea. `anchura` toma como argumento la longitud que indica la anchura de la caja. La principal diferencia entre una `minipage` y una `\parbox` es que usted no puede usar todas las órdenes y entornos dentro de una `parbox`, mientras que casi todo es posible en una `minipage`.

Mientras que `\parbox` empaqueta un párrafo entero partiendo renglones y todo, hay también una clase de órdenes encajonadoras que trabajan sólo con material alineado horizontalmente. Ya conocemos una de ellas; se llama `\mbox`. Simplemente empaqueta una serie de cajas en otra, y puede usarse para impedir a L<sup>A</sup>T<sub>E</sub>X romper dos palabras. Como puede poner cajas dentro de cajas, estos empaquetadores de cajas horizontales le dan total flexibilidad.

La orden

```
\makebox[anchura][pos]{texto}
```

donde `anchura` define la anchura de la caja resultante vista desde fuera,<sup>4</sup> tiene un efecto parecido. Además de las expresiones de longitud, puede también usar `\width`, `\height`, `\depth` y `\totalheight` en el parámetro de anchura. Se establecen a partir de valores obtenidos midiendo el *texto* compuesto. El parámetro `pos` toma una letra como valor: `center` (centro), `flushleft` (izquierda), `flushright` (derecha) o `spread` (expandir el texto hasta llenar la caja).

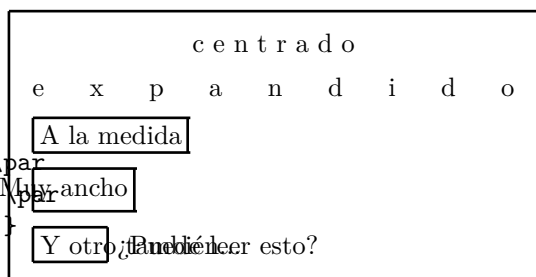
La orden `\framebox` funciona exactamente igual que `\makebox`, pero dibuja una caja alrededor del texto.

El ejemplo siguiente le muestra algunas cosas que podría hacer con las órdenes `\makebox` y `\framebox`.

---

<sup>4</sup>Esto significa que puede ser más pequeña que el material dentro de ella. Usted puede incluso poner la anchura 0pt de forma que el texto de dentro de la caja se componga sin afectar a las cajas de alrededor.

```
\makebox[\textwidth]{%
  c e n t r a d o}\par
\makebox[\textwidth][s]{%
  e x p a n d i d o}\par
\framebox[1.1\width]{A la medida} \par
\framebox[0.8\width][r]{Muy ancho} \par
\framebox[1cm][l]{Y otro también...} \par
¿Puede leer esto?
```



Ahora que controlamos lo horizontal, el siguiente paso obvio es ir por la vertical.<sup>5</sup>

La orden

```
\raisebox{sube}[extiende-sobre-línea-base][extiende-bajo-línea-base]{texto}
```

le permite definir las propiedades verticales de una caja. Puede usar `\width`, `\height`, `\depth` y `\totalheight` en los tres primeros parámetros, para afectar al tamaño de la caja dentro del argumento *texto*.

```
\raisebox{0pt}[0pt][0pt]{\Large%
\textbf{Aaaa}\raisebox{-0.3ex}{a}%
\raisebox{-0.7ex}{aa}%
\raisebox{-1.2ex}{h}%
\raisebox{-2.2ex}{h}%
\raisebox{-4.5ex}{h}}
---gritó, pero ni siquiera el más
próximo se dio cuenta de que
algo terrible le había sucedido...
```

Aaaaaaah —gritó, pero ni siquiera el más próximo se dio cuenta de que algo terrible le había sucedido...

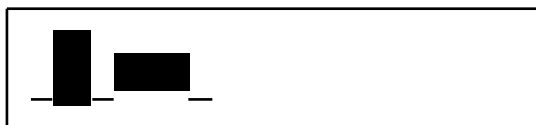
## 6.7. Líneas y puntales

Hace unas páginas puede haber visto la orden

```
\rule[sube]{anchura}{altura}
```

Usada normalmente produce simplemente una caja negra.

```
\rule{3mm}{.1pt}%
\rule[-1mm]{5mm}{1cm}%
\rule{3mm}{.1pt}%
\rule[1mm]{1cm}{5mm}%
\rule{3mm}{.1pt}
```



Esto es útil para dibujar líneas verticales y horizontales. La línea de la página del título, por ejemplo, ha sido creada con una orden `\rule`.

<sup>5</sup>El control total sólo se obtiene controlando tanto lo horizontal como lo vertical...



Un caso especial es una línea sin anchura pero con cierta altura. En composición profesional se llama puntal. Se usa para garantizar que un elemento de una página tiene una cierta altura mínima. Podría usarlo en un entorno `tabular` para asegurarse de que una fila tiene cierta altura mínima.

```
\begin{tabular}{|c|}  
\hline  
\rule{1pt}{4ex}Costeru...\\  
\hline  
\rule{0pt}{4ex}Puntal\  
\hline  
\end{tabular}
```

Costeru...
Puntal

Fin.

# Bibliografía

- [1] Leslie Lamport. *L<sup>A</sup>T<sub>E</sub>X: A Document Preparation System*. Addison-Wesley, Reading, Massachusetts, second edition, 1994, ISBN 0-201-52983-1.
- [2] Donald E. Knuth. *The T<sub>E</sub>Xbook*, Volume A of *Computers and Typesetting*, Addison-Wesley, Reading, Massachusetts, second edition, 1984, ISBN 0-201-13448-9.
- [3] Frank Mittelbach, Michel Goossens, Johannes Braams, David Carlisle, Chris Rowley. *The L<sup>A</sup>T<sub>E</sub>X Companion, (2nd Edition)*. Addison-Wesley, Reading, Massachusetts, 2004, ISBN 0-201-36299-6.
- [4] Michel Goossens, Sebastian Rahtz and Frank Mittelbach. *The L<sup>A</sup>T<sub>E</sub>X Graphics Companion*. Addison-Wesley, Reading, Massachusetts, 1997, ISBN 0-201-85469-4.
- [5] Each L<sup>A</sup>T<sub>E</sub>X installation should provide a so-called *L<sup>A</sup>T<sub>E</sub>X Local Guide*, which explains the things that are special to the local system. It should be contained in a file called `local.tex`. Unfortunately, some lazy sysops do not provide such a document. In this case, go and ask your local L<sup>A</sup>T<sub>E</sub>X guru for help.
- [6] L<sup>A</sup>T<sub>E</sub>X3 Project Team. *L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> for authors*. Comes with the L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> distribution as `usrguide.tex`.
- [7] L<sup>A</sup>T<sub>E</sub>X3 Project Team. *L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> for Class and Package writers*. Comes with the L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> distribution as `clsguide.tex`.
- [8] L<sup>A</sup>T<sub>E</sub>X3 Project Team. *L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> Font selection*. Comes with the L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> distribution as `fntguide.tex`.
- [9] D. P. Carlisle. *Packages in the ‘graphics’ bundle*. Comes with the ‘graphics’ bundle as `grfguide.tex`, available from the same source your L<sup>A</sup>T<sub>E</sub>X distribution came from.
- [10] Rainer Schöpf, Bernd Raichle, Chris Rowley. *A New Implementation of L<sup>A</sup>T<sub>E</sub>X’s verbatim Environments*. Comes with the ‘tools’ bundle as

`verbatim.dtx`, available from the same source your  $\text{\LaTeX}$  distribution came from.

- [11] Vladimir Volovich, Werner Lemberg and  $\text{\LaTeX}$ 3 Project Team. *Cyrillic languages support in  $\text{\LaTeX}$* . Comes with the  $\text{\LaTeX}$  2 $\epsilon$  distribution as `cyrguide.tex`.
- [12] Graham Williams. *The TeX Catalogue* is a very complete listing of many  $\text{\TeX}$  and  $\text{\LaTeX}$  related packages. Available online from [CTAN: /tex-archive/help/Catalogue/catalogue.html](http://ctan.org/tex-archive/help/Catalogue/catalogue.html)
- [13] Keith Reckdahl. *Using EPS Graphics in  $\text{\LaTeX}$  2 $\epsilon$  Documents*, which explains everything and much more than you ever wanted to know about EPS files and their use in  $\text{\LaTeX}$  documents. Available online from [CTAN:/tex-archive/info/epslatex.ps](http://ctan.org/tex-archive/info/epslatex.ps)
- [14] Kristoffer H. Rose. *Xy-pic User's Guide*. Downloadable from CTAN with Xy-pic distribution
- [15] John D. Hobby. *A User's Manual for METAPOST*. Downloadable from <http://cm.bell-labs.com/who/hobby/>
- [16] Alan Hoenig. *TeX Unbound*. Oxford University Press, 1998, ISBN 0-19-509685-1; 0-19-509686-X (pbk.)
- [17] Urs Oswald. *Graphics in  $\text{\LaTeX}$  2 $\epsilon$* , containing some Java source files for generating arbitrary circles and ellipses within the `picture` environment, and *METAPOST- A Tutorial*. Both downloadable from <http://www.ursoswald.ch>
- [18] Till Tantau. *TikZ&PGF Manual*. Download from [CTAN:/tex-archive/graphics/pgf/base/doc/generic/pgf/pgfmanual.pdf](http://ctan.org/tex-archive/graphics/pgf/base/doc/generic/pgf/pgfmanual.pdf)
- [19] François Charette. *Polyglossia: A Babel Replacement for  $\text{\TeX}$* . Comes with the  $\text{\TeX}$ Live distribution as `polyglossia.pdf`. (Type `texdoc polyglossia` on the command line.)
- [20] François Charette. *An ArabTeX-like interface for typesetting languages in Arabic script with  $\text{\TeX}$* . Comes with the  $\text{\TeX}$ Live distribution as `arabxetex.pdf`. (Type `texdoc arabxetex` on the command line.)
- [21] Will Robertson and Khaled Hosny. *The fontspec package*. Comes with the  $\text{\TeX}$ Live distribution as `fontspec.pdf`. (Type `texdoc fontspec` on the command line.)
- [22] Apostolos Syropoulos. *The xgreek package*. Comes with the  $\text{\TeX}$ Live distribution as `xgreek.pdf`. (Type `texdoc xgreek` on the command line.)

- 
- [23] Vafa Khalighi. *The  `bidi`  package*. Comes with the T<sub>E</sub>XLive distribution as  `bidi.pdf` . (Type  `texdoc bidi`  on the command line.
  - [24] Vafa Khalighi. *The  `XePersian`  package*. Comes with the T<sub>E</sub>XLive distribution as  `xepersian-doc.pdf` . (Type  `texdoc xepersian`  on the command line.
  - [25] Wenchang Sun. *The  `xeCJK`  package*. Comes with the T<sub>E</sub>XLive distribution as  `xeCJK.pdf` . (Type  `texdoc xecjk`  on the command line.

# Índice alfabético

## Symbols

$\backslash!$ , 58  
", 21  
\$, 51  
 $\backslash($ , 51  
 $\backslash)$ , 51  
 $\backslash,$ , 52, 58  
-, 22  
—, 22  
 $\backslash-$ , 20  
—, 22  
—, 22  
., espacio tras, 37  
..., 23  
 $\backslash:$ , 58  
 $\backslash;$ , 58  
 $\backslash@$ , 37  
 $\backslash[$ , 52  
índice, 77  
índice general, 38  
órdenes, 5  
órdenes frágiles, 49  
L<sup>A</sup>T<sub>E</sub>Xteam, 1  
 $\backslash\backslash$ , 19, 42, 43, 45, 124  
 $\backslash\backslash*$ , 19  
fundición, 117  
 $\backslash]$ , 52  
~, 37

## A

A4 papel, 11  
A5 papel, 11  
å, 25  
abstract, 43  
acento, 24  
Acrobat Reader, 82

$\backslash$ addtolength, 126  
æ, 25  
aiguill, 83  
agrupar, 118  
agudo, 25  
alemán, 26  
alineado, 42  
amsbsy, 64  
amsmath, 53, 72  
amsmath, 52, 56–58, 60, 61, 64  
amssymb, 53, 65  
amsthm, 62, 63  
 $\backslash$ and, 39  
ansinew, 27  
antibarra, 5  
 $\backslash$ appendix, 38, 39  
applemac, 27  
Arabic, 36  
arabxetex, 36  
 $\backslash$ arccos, 55  
 $\backslash$ arcsen, 55  
 $\backslash$ arcsin, 55  
 $\backslash$ arctan, 55  
 $\backslash$ arctg, 55  
 $\backslash$ arg, 55  
array, 59  
article clase, 10  
 $\backslash$ author, 39, 87

## B

B5 papel, 11  
babel, 20, 25, 26, 34, 109  
 $\backslash$ backmatter, 39  
 $\backslash$ backslash, 5  
beamer, 10  
beamer, 92–94

\chapter, 38

- 
- `\chaptermark`, 78
  - `\ci`, 113
  - `\circle`, 102
  - `\circle*`, 102
  - `\cite`, 75
  - `\cleardoublepage`, 49
  - `\clearpage`, 49
  - `\cline`, 45
  - `\cos`, 55
  - `\cosec`, 55
  - `\cosh`, 55
  - `\cot`, 55
  - `\cotg`, 55
  - `\coth`, 55
  - `\csc`, 55
  - `\date`, 39
  - `\ddots`, 58
  - `\deg`, 55
  - `\depth`, 128, 129
  - `\det`, 55
  - `\dim`, 55
  - `\displaystyle`, 61
  - `\documentclass`, 9, 12, 20
  - `\dum`, 113
  - `\emph`, 41, 118
  - `\end`, 41, 98
  - `\eqref`, 52
  - `\EUR`, 23
  - `\EURtm`, 23
  - `\exp`, 55
  - `\fbox`, 21
  - `\foldera`, 106
  - `\folderb`, 106
  - `\footnote`, 40, 49
  - `\footskip`, 125
  - `\frac`, 56
  - `\framebox`, 128
  - `\frenchspacing`, 37
  - `\frontmatter`, 39
  - `\fussy`, 20
  - `\gcd`, 55
  - `\headheight`, 125
  - `\headsep`, 125
  - `\height`, 128, 129
  - `\hline`, 45
  - `\hom`, 55
  - `\href`, 87, 90
  - `\hspace`, 115, 122
  - `\hyphenation`, 20
  - `\idotsint`, 58
  - `\ifpdf`, 89
  - `\ignorespaces`, 115
  - `\ignorespacesafterend`, 115
  - `\iiiint`, 58
  - `\iiint`, 58
  - `\iint`, 58
  - `\include`, 14, 15
  - `\includegraphics`, 74, 85, 89, 127
  - `\includeonly`, 15
  - `\indent`, 122
  - `\index`, 77, 78
  - `\inf`, 55
  - `\input`, 15
  - `\int`, 56
  - `\intc`, 114
  - `\item`, 42
  - `\ker`, 55
  - `\label`, 40, 52
  - `\LaTeX`, 21
  - `\LaTeXe`, 21
  - `\ldots`, 23, 57
  - `\left`, 57
  - `\leftmark`, 78
  - `\lg`, 55
  - `\lim`, 55
  - `\liminf`, 55
  - `\limsup`, 55
  - `\line`, 100, 106
  - `\linebreak`, 19
  - `\linespread`, 121
  - `\linethickness`, 103, 104, 106
  - `\listoffigures`, 48
  - `\listoftables`, 48
  - `\ln`, 55
  - `\log`, 55
  - `\mainmatter`, 39, 88
  - `\makebox`, 128
  - `\Makefile`, 116

\makeindex, 77  
\maketitle, 39  
\marginparpush, 125  
\marginparsep, 125  
\marginparwidth, 125  
\mathbb, 53  
\mathrm, 61  
\max, 55  
\mbox, 21, 24, 128  
\min, 55  
\multicolumn, 46  
\multiput, 99, 103  
\newcommand, 114  
\newenvironment, 115  
\newline, 19  
\newpage, 19  
\newsavebox, 105  
\newtheorem, 62  
\newtheoremstyle, 62  
\noindent, 122  
\nolinebreak, 19  
\nonumber, 60  
\nopagebreak, 19  
\not, 66  
\oddsidemargin, 125  
\oval, 104, 106  
\overbrace, 54  
\overleftarrow, 55  
\overline, 54  
\overrightarrow, 55  
\pagebreak, 19  
\pageref, 40, 81  
\pagestyle, 12  
\paperheight, 125  
\paperwidth, 125  
\par, 118  
\paragraph, 38  
\parbox, 128  
\parindent, 122  
\parskip, 122  
\part, 38  
\phantom, 49, 60  
\pmod, 55  
\Pr, 55  
\printindex, 78  
\prod, 56  
\protect, 49  
\providecommand, 114  
\ProvidesPackage, 117  
\put, 99–105  
\qbezier, 97, 99, 106  
\qedhere, 63  
\qquad, 52, 58  
\quad, 52, 58  
\raisebox, 129  
\ref, 40, 52, 81  
\renewcommand, 114  
\renewenvironment, 115  
\right, 57, 59  
\right., 57  
\rightmark, 78  
\rule, 115, 129  
\savebox, 105  
\scriptscriptstyle, 61  
\scriptstyle, 61  
\sec, 55  
\section, 38, 49  
\sectionmark, 78  
\selectlanguage, 26  
\sen, 55  
\senh, 55  
\setlength, 98, 122, 126  
\settodepth, 126  
\settoheight, 126  
\settowidth, 126  
\shorthandoff, 109  
\sin, 55  
\sinh, 55  
\slash, 22  
\sloppy, 20  
\smallskip, 124  
\sqrt, 54  
\stackrel, 56  
\stretch, 115, 123  
\subparagraph, 38  
\subsection, 38  
\subsectionmark, 78  
\substack, 56



- `\subsubsection`, 38
- `\sum`, 56
- `\sup`, 55
- `\tableofcontents`, 38
- `\tan`, 55
- `\tanh`, 55
- `\TeX`, 21
- `\texorpdfstring`, 88, 89
- `\textcelsius`, 23
- `\texteuro`, 23
- `\textheight`, 125
- `\textrm`, 61
- `\textstyle`, 61
- `\textwidth`, 125
- `\tg`, 55
- `\tgh`, 55
- `\thicklines`, 101, 104, 106
- `\thinlines`, 104, 106
- `\thispagestyle`, 12
- `\title`, 39
- `\today`, 21
- `\topmargin`, 125
- `\totalheight`, 128, 129
- `\underbrace`, 54
- `\underline`, 41, 54
- `\unitlength`, 98, 100
- `\usebox`, 105
- `\usepackage`, 10, 12, 23, 26, 27, 117
- `\usetikzlibrary`, 110
- `\vdots`, 58
- `\vec`, 55
- `\vector`, 101
- `\verb`, 44
- `\verbatiminput`, 80
- `\vspace`, 123
- `\widehat`, 55
- `\widetilde`, 55
- `\width`, 128, 129
- comment, 6
- comment, 6
- corchetes, 5
- `\cos`, 55
- `\cosec`, 55
- `\cosh`, 55
- `\cot`, 55
- `\cotg`, 55
- `\coth`, 55
- cp1251, 27
- cp850, 27
- cp866nav, 27
- `\csc`, 55
- cursiva, 118
- Cyrillic, 35
- D**
- `\date`, 39
- dcolumn, 46
- `\ddots`, 58
- decimal alignment, 46
- `\deg`, 55
- delimitadores, 57
- `\depth`, 128, 129
- description, 42
- deslizantes, elementos, 47
- `\det`, 55
- `\dim`, 55
- dimensiones, 122
- `\displaymath`, 52
- `\displaystyle`, 61
- doble espaciado de renglones, 121
- doc, 13
- `\documentclass`, 9, 12, 20
- dos caras, 11
- dos columnas, 11
- `\dum`, 113
- E**
- ecuaciones largas, 60
- eepic, 97, 102
- `\emph`, 41, 118
- empty, 12
- en blanco, 4
- Encapsulated POSTSCRIPT, 73, 85
- encodings
  - font
    - LGR, 27
    - OT1, 27

- T1, 27
- T2A, 27
- T2B, 27
- T2C, 27
- X2, 27
- input
  - ansinew, 27
  - applemac, 27
  - cp1251, 27
  - cp850, 27
  - cp866nav, 27
  - koi8-ru, 27
  - latin1, 27
  - macukr, 27
  - utf-8, 27
  - utf8, 27
- \end, 41, 98
- enumerate, 42
- environments
  - abstract, 43
  - array, 59
  - block, 95
  - center, 42
  - comment, 6
  - description, 42
  - displaymath, 52
  - enumerate, 42
  - eqnarray, 59
  - equation, 52
  - figure, 47, 48
  - flushleft, 42
  - flushright, 42
  - frame, 93, 95
  - itemize, 42
  - lscommand, 113
  - math, 51
  - minipage, 128
  - parbox, 128
  - picture, 97, 98, 102, 103
  - proof, 63
  - pspicture, 98
  - quotation, 43
  - quote, 43
  - subarray, 56
  - table, 47, 48
  - tabular, 44, 127
  - thebibliography, 75
  - tikzpicture, 108
  - verbatim, 44, 80
  - verse, 43
- epic, 97
- eqnarray, 59
- \eqref, 52
- equation, 52
- escandinavas letras, 25
- español, 28
- espaciado matemático, 58
- espacio, 4
- espacio en blanco
  - al principio de línea, 4
  - tras órdenes, 5
- espacio entre renglones, 121
- estilos de página, 12
- estructura, 7
- \EUR, 23
- eurosym, 23
- \EURtm, 23
- executive papel, 11
- \exp, 55
- exponent, 54
- exscale, 13, 57
- extension
  - .aux, 14
  - .cls, 12
  - .dtx, 12
  - .dvi, 14, 74
  - .eps, 74
  - .fd, 12
  - .idx, 14, 77
  - .ilg, 14
  - .ind, 14, 78
  - .ins, 12
  - .lof, 14
  - .log, 14
  - .lot, 14
  - .sty, 12, 80
  - .tex, 8, 12
  - .toc, 14

extensiones, 12

## F

fancyhdr, 78, 79

\fbox, 21

fichero de entrada, 7

figure, 47, 48

flecha, 55

flushleft, 42

flushright, 42

foiltex, 10

\foldera, 106

\folderb, 106

font

\footnotesize, 118

\Huge, 118

\huge, 118

\LARGE, 118

\Large, 118

\large, 118

\mathbf, 119

\mathcal, 119

\mathit, 119

\mathnormal, 119

\mathrm, 119

\mathsf, 119

\mathtt, 119

\normalsize, 118

\scriptsize, 118

\small, 118

\textbf, 118

\textit, 118

\textmd, 118

\textnormal, 118

\textrm, 118

\textsc, 118

\textsf, 118

\textsl, 118

\texttt, 118

\textup, 118

\tiny, 118

font encodings

LGR, 27

OT1, 27

T1, 27

T2A, 27

T2B, 27

T2C, 27

X2, 27

font size, 118

fontenc, 13, 27

fontspec, 34, 91, 92

\footnote, 40, 49

\footnotesize, 118

\footskip, 125

formulae, 51

\frac, 56

fracción, 56

frame, 93, 95

\framebox, 128

\frenchspacing, 37

\frontmatter, 39

\fussy, 20

## G

\gcd, 55

geometry, 80

GhostScript, 73

gráficos, 10, 73

grado, 23

graphicx, 73, 85, 93

grave, 25

Greek, 35

grupos, 118

guión, 22

## H

\headheight, 125

texttttheadings, 12

\headsep, 125

Hebrew, 36

\height, 128, 129

hipertexto, 81

\hline, 45

\hom, 55

horizontal

brace, 54

espacio, 122

- línea, 54
- \href, 87, 90
- \hspace, 115, 122
- \Huge, 118
- \huge, 118
  - hyperref, 36, 82, 85, 89, 90, 93
  - hyphenat, 80
- \hyphenation, 20
- I**
  - i y j sin punto (i y j), 25
  - idioma, 25
  - \idotsint, 58
  - ifpdf, 89
  - \ifpdf, 89
  - ifthen, 13
  - \ignorespaces, 115
  - \ignorespacesafterend, 115
  - \iiiint, 58
  - \iiint, 58
  - \iint, 58
  - \include, 14, 15
  - \includegraphics, 74, 85, 89, 127
  - \includeonly, 15
  - \indent, 122
    - indentfirst, 122
  - \index, 77, 78
  - \inf, 55
  - \input, 15
    - input encodings
      - ansinew, 27
      - applemac, 27
      - cp1251, 27
      - cp850, 27
      - cp866nav, 27
      - koi8-ru, 27
      - latin1, 27
      - macukr, 27
      - utf-8, 27
      - utf8, 27
    - inputenc, 13, 26
  - \int, 56
  - \intc, 114
  - integral, 56
  - international, 25
  - \item, 42
  - itemize, 42
  - J**
    - Japanese, 37
    - Jawi, 36
  - K**
    - kashida, 36
    - Kashmiri, 36
    - \ker, 55
    - Knuth, Donald E., 1
    - koi8-ru, 27
    - Korean, 37
    - Kurdish, 36
  - L**
    - \label, 40, 52
    - Lamport, Leslie, 1
    - \LARGE, 118
    - \Large, 118
    - \large, 118
    - \LaTeX, 21
    - L<sup>A</sup>T<sub>E</sub>X3, 4
    - \LaTeXe, 21
    - latexsym, 13
    - latin1, 27
    - layout, 124
    - \ldots, 23, 57
    - \left, 57
    - \leftmark, 78
    - legal papel, 11
    - letras griegas, 54
    - letter papel, 11
    - \lg, 55
    - LGR, 27
    - ligadura, 24
    - \lim, 55
    - \liminf, 55
    - \limsup, 55
    - \line, 100, 106
    - \linebreak, 19
    - \linespread, 121
    - \linethickness, 103, 104, 106

\listoffigures, 48

\listoftables, 48

llaves, 5, 57, 118

\ln, 55

\log, 55

longtable, 46

lscommand, 113

## M

márgenes, 124

módulo, 55

macukr, 27

\mainmatter, 39, 88

\makebox, 128

\Makefile, 116

makeidx, 13, 77

makeidx, paquete, 77

\makeindex, 77

makeindex, programa, 77

\maketitle, 39

Malay, 36

\marginparpush, 125

\marginparsep, 125

\marginparwidth, 125

marvosym, 23

matemáticas, 51

matemático

acento, 54

delimitador, 57

menos, 22

math, 51

\mathbb, 53

\mathbf, 119

\mathcal, 119

mathematical

functions, 55

\mathit, 119

\mathnormal, 119

\mathrm, 61, 119

mathrsfs, 72

\mathsf, 119

\mathtt, 119

\max, 55

\mbox, 21, 24, 128

METAPOST, 85

microtype, 92

\min, 55

minimal clase, 10

minipage, 128

mltex, 83

mltex, 83

\multicolumn, 46

\multirow, 99, 103

## N

negrita, 53, 118

negrita de pizarra, 53

\newcommand, 114

\newenvironment, 115

\newline, 19

\newpage, 19

\newsavebox, 105

\newtheorem, 62

\newtheoremstyle, 62

\noindent, 122

\nolinebreak, 19

\nonumber, 60

\nopagebreak, 19

\normalsize, 118

\not, 66

## O

oblicua, 118

\oddsidemargin, 125

œ, 25

opciones, 9

OT1, 27

Ottoman, 36

\oval, 104, 106

\overbrace, 54

overfull hbox, 20

\overleftarrow, 55

\overline, 54

\overrightarrow, 55

## P

página

composición, 124

párrafo, 17

- package, 10
- packages
  - aeguill, 83
  - ambsy, 64
  - amsfonts, 53, 72
  - amsmath, 52, 56–58, 60, 61, 64
  - amssymb, 53, 65
  - amsthm, 62, 63
  - arabxetex, 36
  - babel, 20, 25, 26, 34, 109
  - beamer, 92–94
  - bidir, 35, 36
  - bm, 64
  - calc, 126
  - color, 93
  - comment, 6
  - dcolum, 46
  - doc, 13
  - eepic, 97, 102
  - epic, 97
  - eurosym, 23
  - exscale, 13, 57
  - fancyhdr, 78, 79
  - fontenc, 13, 27
  - fontspec, 34, 91, 92
  - geometry, 80
  - graphicx, 73, 85, 93
  - hyperref, 36, 82, 85, 89, 90, 93
  - hyphenat, 80
  - ifpdf, 89
  - ifthen, 13
  - indentfirst, 122
  - inputenc, 13, 26
  - latexsym, 13
  - layout, 124
  - longtable, 46
  - makeidx, 13, 77
  - marvosym, 23
  - mathrsfs, 72
  - microtype, 92
  - mltex, 83
  - pgf, 108, 111
  - pgfplot, 111
  - polyglossia, 34–36
  - ppower4, 93
  - prosper, 93
  - pstricks, 97, 98, 102
  - pxfonts, 84
  - showidx, 78
  - syntonly, 13, 15
  - textcomp, 23
  - TikZ, 109
  - tikz, 108
  - txfonts, 84
  - verbatim, 6, 80
  - xeCJK, 37
  - xepersian, 36
  - xgreek, 35
- page style
  - empty, 12
  - headings, 12
  - plain, 12
- \pagebreak, 19
- \pageref, 40, 81
- \pagestyle, 12
- Palabra, 78
- palo seco, 118
- papel
  - tamaño, 82
- \paperheight, 125
- \paperwidth, 125
- paquete, 7, 113
- \par, 118
- parámetro, 5
- parámetros opcionales, 5
- \paragraph, 38
- \parbox, 128
- parbox, 128
- \parindent, 122
- \parskip, 122
- \part, 38
- Pashto, 36
- PDF, 81, 90
- PDF<sub>L</sub>AT<sub>E</sub>X, 93
- pdf<sub>L</sub>AT<sub>E</sub>X, 83, 92
- pdf<sub>L</sub>AT<sub>E</sub>X, 82
- pdf<sub>T</sub>E<sub>X</sub>, 82
- Persian, 36

pgf, 108, 111  
 pgfplot, 111  
 \phantom, 49, 60  
 picture, 97, 98, 102, 103  
 pies de página, 12  
 plain, 12  
 \pmod, 55  
 polyglossia, 34–36  
 POSTSCRIPT  
   Encapsulated, 73, 85  
 POSTSCRIPT, 9, 49, 73, 74, 83, 84, 98  
 ppower4, 93  
 \Pr, 55  
 preámbulo, 7  
 prima, 55  
 \printindex, 78  
 proc clase, 10  
 \prod, 56  
 productorio, 56  
 proof, 63  
 prosper, 10  
 prosper, 93  
 \protect, 49  
 \providecommand, 114  
 \ProvidesPackage, 117  
 pspicture, 98  
 pstricks, 97, 98, 102  
 puntal, 130  
 punto, 23  
 puntos diagonales, 58  
 puntos horizontales, 58  
 puntos suspensivos, 23  
 puntos verticales, 58  
 \put, 99–105  
 pxfonts, 84

## Q

\qbezier, 97, 99, 106  
 \qedhere, 63  
 \qqquad, 52, 58  
 \quad, 52, 58  
 quotation, 43  
 quote, 43

## R

raíz cuadrada, 54  
 \raisebox, 129  
 raya, 22  
 raya corta, 22  
 recta, 118  
 \ref, 40, 52, 81  
 referencias cruzadas, 40  
 rematada, 118  
 \renewcommand, 114  
 \renewenvironment, 115  
 report clase, 10  
 \right, 57, 59  
 \right., 57  
 \rightmark, 78  
 \rule, 115, 129  
 Russian, 35

## S

símbolos en negrita, 64  
 saltos de línea, 19  
 \savebox, 105  
 \scriptscriptstyle, 61  
 \scriptsize, 118  
 \scriptstyle, 61  
 \sec, 55  
 \section, 38, 49  
 \sectionmark, 78  
 \selectlanguage, 26  
 \sen, 55  
 \senh, 55  
 \setlength, 98, 122, 126  
 \settodepth, 126  
 \settoheight, 126  
 \settowidth, 126  
 \shorthandoff, 109  
 showidx, 78  
 signo menos, 22  
 \sin, 55  
 Sindhi, 36  
 \sinh, 55  
 sistemas de ecuaciones, 59  
 Slash, 22  
 \slash, 22

- slides clase, 10
- \sloppy, 20
- \small, 118
- \smallskip, 124
- \sqrt, 54
- \stackrel, 56
- \stretch, 115, 123
  - subarray, 56
- \subparagraph, 38
  - subscript, 54
- \subsection, 38
- \subsectionmark, 78
- \substack, 56
- \subsubsection, 38
- \sum, 56
  - sumatorio, 56
- \sup, 55
  - superíndice, 56
- syntonly, 13, 15
- T**
- T1, 27
- T2A, 27
- T2B, 27
- T2C, 27
- título, 11, 39
- título del documento, 11
- tabla, 44
- table, 47, 48
- \tableofcontents, 38
- tabular, 44, 127
- tamaño de fundición básico, 11
- tamaño de fundición del documentd,  
11
- tamaño de fundición en matemático,  
61
- tamaño de la fundición, 117
- tamaño del papel, 11, 124
- \tan, 55
- \tanh, 55
- \TeX, 21
- \texorpdfstring, 88, 89
- \textbf, 118
- \textcelsius, 23
- textcomp, 23
- \texteuro, 23
- \textheight, 125
- \textit, 118
- \textmd, 118
- \textnormal, 118
  - texto en color, 10
- \textrm, 61, 118
- \textsc, 118
- \textsf, 118
- \textsl, 118
- \textstyle, 61
- \texttt, 118
- \textup, 118
- \textwidth, 125
- \tg, 55
- \tgh, 55
  - thebibliography, 75
- \thicklines, 101, 104, 106
- \thinlines, 104, 106
- \thispagestyle, 12
- TikZ, 109
- tikz, 108
- tikzpicture, 108
- tilde, 22, 54
- tilde ( ~), 37
- \tiny, 118
  - tipos de fichero, 12
- \title, 39
- \today, 21
- \topmargin, 125
- \totalheight, 128, 129
- tres puntos, 57
- Turkish, 36
- txfonts, 84
- U**
- Uighur, 36
- umlaut, 25
- una cara, 11
- una columna, 11
- \underbrace, 54
  - underfull hbox, 20
- \underline, 41, 54



- unidades, 122, 123
- `\unitlength`, 98, 100
- Urdu, 36
- URL, 22
- `\usebox`, 105
- `\usepackage`, 10, 12, 23, 26, 27, 117
- `\usetikzlibrary`, 110
- utf-8, 27
- utf8, 27

## V

- `\vdots`, 58
- `\vec`, 55
- `\vector`, 101
- vectors, 55
- ventajas de L<sup>A</sup>T<sub>E</sub>X, 3
- `\verb`, 44
- verbatim, 6, 80
- verbatim, 44, 80
- `\verbatiminput`, 80
- Versalitas, 118
- verse, 43
- vertical
  - espacio, 123
- `\vspace`, 123

## W

- `\widehat`, 55
- `\widetilde`, 55
- `\width`, 128, 129
- www, 22
- WYSIWYG, 2, 3

## X

- x2, 27
- xeCJK, 37
- X<sub>Ǝ</sub>L<sup>A</sup>T<sub>E</sub>X, 90
- xepersian, 36
- X<sub>Ǝ</sub>T<sub>E</sub>X, 90
- xgreek, 35
- Xpdf, 82





# Aprende git

Pablo Hinojosa y JJ Merelo

## Licencia

Esta obra está sujeta a la licencia Reconocimiento-CompartirIgual 4.0 Internacional de Creative Commons. Para ver una copia de esta licencia, visite <http://creativecommons.org/licenses/by-sa/4.0/>.



Figure 1: cc-by-sa

## Prólogo y agradecimientos

Este libro se escribió originalmente para el [curso de git](#) impartido en el Centro de Enseñanzas Virtuales de la UGR en abril y mayo de 2014. Por lo tanto, agradecemos a los alumnos la retroalimentación ofrecida, especialmente a Manuel Cogolludo, que se revisó el material y nos hizo grandes sugerencias.

También es una primera edición, por lo que seguramente habrá algún (ciento de) fallo. Pero dado que es un libro libre, puedes corregir esos fallos y hacerte tu propia versión en GitHub o [hacernos sugerencias y comentarios](#) que atenderemos rápida y debidamente. Si te atreves una vez escrito el libro a hacer los cambios tú mismo y a solicitarnos la incorporación a versiones posteriores del libro, te lo agradeceremos aún más y aparecerás en este prólogo.

El libro está dirigido a una persona que no sepa programar, o que sepa sin que haga falta ningún lenguaje específico. Sí hace falta cierto manejo de la línea de órdenes y el uso de sistemas operativos adecuados para el desarrollo, lo que probablemente usas de todas formas si estás acostumbrado a usar la línea de órdenes. Sólo al final hay algunos programas con la idea de que se aprenda a modificarlos para usarlos en git, no tanto a dominarlos. En general, git va bien con cualquier lenguaje de programación.

## Agradecimientos adicionales

Muchas personas han contribuido con [correcciones o aportaciones y están todas ahí, en el gráfico de GitHub](#) pero está bien nombrarlas; agradecemos a Miguel Ángel Pedregosa, Alfonso Romero y a [Juanmi](#), [Fernando Tricas](#) y [Diego](#) su revisión extensiva de los fuentes y las correcciones aportadas.

## Introducción

### Objetivos

- Saber qué es el Software Libre
- Aprender qué es un Sistema de Control de Versiones
- Ver los Sistema de Control de Versiones y sus tipos
- Conocer las principales características de git

### Software Libre

En los últimos años, el software libre se ha ido extendiendo hasta abarcar la mayor parte de sistemas de Internet, supercomputadores o servicios de red, llegando finalmente a ordenadores personales, tablets, teléfonos e incluso electrodomésticos.

Llamamos [Software Libre](#) al que permite al usuario ejercer una serie de libertades, a saber:

- la libertad de usar el programa, con cualquier propósito.
- la libertad de estudiar cómo funciona el programa y modificarlo, adaptándolo a tus necesidades.
- la libertad de distribuir copias del programa, con lo cual puedes ayudar a tu prójimo.
- la libertad de mejorar el programa y hacer públicas esas mejoras a los demás, de modo que toda la comunidad se beneficie.

Para que un programa sea libre hacen falta dos requisitos fundamentales:

- Una licencia que permita el uso, modificación y distribución del programa
- Acceso al código fuente del programa

Además, en la comunidad de software libre se dan algunas peculiaridades, como la internacionalización o el trabajo colaborativo de gran cantidad de desarrolladores, que han promovido la creación de forjas, repositorios y sistemas que permitan compartir ese código de forma abierta y además, administrar y gestionar todo ese trabajo de una forma eficiente.

git es uno de estos sistemas, y es software libre.

### Sistemas de control de Versiones

En cualquier proyecto de desarrollo es necesario gestionar los cambios, modificaciones, añadidos etc. que se han hecho a lo largo de la historia de dicho proyecto.

Si se trata de un trabajo pequeño, de corta duración y es llevado a cabo por un solo programador, quizás un archivo histórico de copias de seguridad pueda ser suficiente, pero esto se vuelve claramente insuficiente al crecer la complejidad del proyecto.

Algunos de los problemas más habituales a los que se enfrenta cualquier persona que participe en un desarrollo informático son:

- Dos o más programadores modifican el mismo archivo de código y hay que gestionar ese conflicto.
- Es necesario mantener (al menos) dos versiones del proyecto, una en producción y otra en desarrollo.
- Algún cambio ha sido desechado y es necesario regresar a una fase anterior del proyecto.
- Se inicia un “fork” o proyecto derivado a partir del estado actual del proyecto.

Llevar a cabo la administración de estos aspectos (y de muchos otros) manualmente es materialmente imposible, y es para ello para lo que se inventaron los Sistemas de Control de Versiones.

Debido al crecimiento en extensión y complejidad del software los Sistemas de Control de Versiones están tomando cada vez más importancia. En especial, los proyectos de software libre, que tienden a aunar los esfuerzos de un gran número de programadores trabajando simultáneamente en múltiples versiones del código, han estado a la vanguardia del uso de estas herramientas.

Estos sistemas nacieron para gestionar código fuente, pero pueden ser usados para cualquier tipo de documentación (aunque no pueden aprovechar todo su potencial en archivos que no sean de texto plano). Por ejemplo, este curso ha sido desarrollado bajo git y su código puede encontrarse en su repositorio oficial en [github.com/oslugr/curso-git](https://github.com/oslugr/curso-git)

## Tipos de Sistemas de Control de Versiones

Existen muchos Sistemas de Control de Versiones como [CVS](#), [Subversion](#), [Bazaar](#), [Mercurial](#) o, por supuesto, [Git](#), pero todos pueden clasificarse en dos tipos fundamentales según su modo de trabajo.

**Sistemas centralizados** Los sistemas de control de versiones centralizados fueron los primeros en aparecer y son los de funcionamiento más simple e intuitivo.

En ellos, existe un repositorio central donde se archiva el código y toda la información asociada (marchas de tiempo, autores de los cambios, etc). Los distintos desarrolladores trabajan con copias de ese código que actualizan a partir del servidor central, a donde también envían sus cambios.

Es decir, la versión “oficial” de referencia es la que hay en ese repositorio, y todos los programadores sincronizan sus versiones con esa.

Los programas más conocidos de este tipo son CVS y Subversion.

**Sistemas distribuidos.** Los sistemas distribuidos son más complejos, pero a cambio ofrecen una mayor flexibilidad.

En estos sistemas no existe un servidor central, sino que cada programador tiene su propio repositorio que puede sincronizar con el de cada uno de los demás.

Hay que hacer notar que, aunque no es necesario en este tipo de arquitectura, en la práctica se suele definir un repositorio de referencia para albergar la versión “oficial” del software.

Los sistemas distribuidos más conocidos son Bazaar y, por supuesto, git.

## git

Git nació para ser el Sistema de Control de Versiones del kernel de Linux (de hecho, fue originalmente programado por el mismo *Linus Torvalds*) y por ello está preparado para proyectos grandes, con muchos desarrolladores y un gran número de ramas.

Se trata, como ya hemos dicho, de un Sistema de Control de Versiones distribuido, por lo que cada programador cuenta con su propio repositorio. Esto hace que, salvo cuando llega el momento de sincronizar con otro repositorio, todo el trabajo se haga en local, con ventajas como la velocidad o que se pueda trabajar sin acceso a la red.

git es software libre, y su código está disponible en su [repositorio de GitHub](#).

El sitio oficial de git es [git-scm.com](https://git-scm.com)

Git es también multiplataforma, y existen versiones para Linux, Mac, Windows y Solaris.

En este curso se usará Linux para los ejemplos y referencias, y se recomienda encarecidamente su uso. Otros sistemas operativos no están tan preparador para algunas tareas como administrar sesiones remotas, etc.

En cualquier caso, el uso del propio git en todos ellos es similar por lo que, salvo las particularidades y limitaciones que pueda tener cada uno, no hay ningún problema a la hora de seguir este curso desde otro sistema operativo.

## Abrir una cuenta en Github

Más adelante en este curso se hablará de Github y se darán detalles sobre su uso pero, por ahora, será suficiente para nosotros con saber abrir una cuenta (que usaremos para enviar nuestros ejercicios).

Desde la propia [página principal de la web de GitHub](#) y como cualquier otro registro, se nos solicitan sólo tres datos: nombre o nick, e-mail y contraseña:

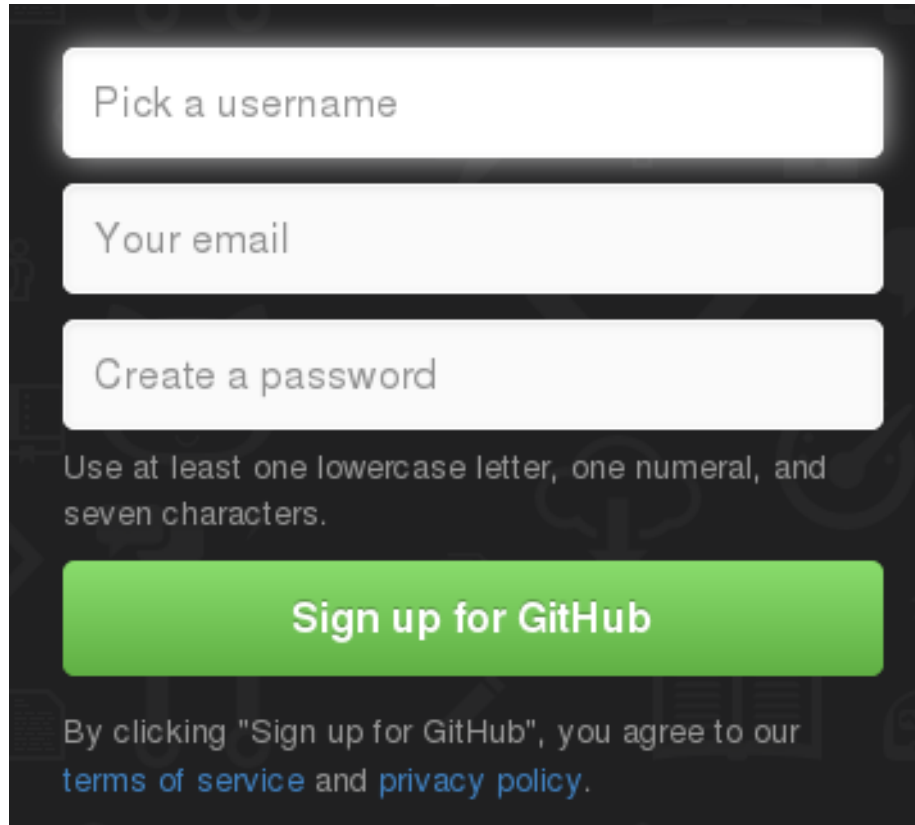
The image shows the GitHub sign-up interface. It features three white input fields on a dark background. The first field is labeled 'Pick a username', the second 'Your email', and the third 'Create a password'. Below the password field, there is a text requirement: 'Use at least one lowercase letter, one numeral, and seven characters.' A prominent green button with the text 'Sign up for GitHub' is positioned below the requirements. At the bottom, a line of text states: 'By clicking "Sign up for GitHub", you agree to our [terms of service](#) and [privacy policy](#).'

Figure 2: Formulario de GitHub

## Uso básico de git

### Objetivos

En este apartado veremos cómo se usa git de forma básica para trabajar en modo monousuario y con un repositorio centralizado.



- Instalar Git
- Crear un repositorio
- Mantener un control de cambios sobre nuestros archivos
- Sincronizar dos o más repositorios
- Tareas básicas de git

## Instalar git

`git` es software libre y se puede instalar en cualquier sistema operativo. A continuación los más populares, empezando por el que aconsejamos para desarrollar software en general, Linux.

**En Linux** Instalar git en Linux es tan simple como usar tu gestor de paquetes favorito. Por ejemplo (recuerda que normalmente necesitarás privilegios de *root* para instalar cualquier programa):

**En Arch Linux** `# pacman -S git`

**En sistemas Debian, Ubuntu, Mint...** `# apt-get install git`

**En Gentoo** `# emerge --ask --verbose dev-vcs/git`

**En sistemas Red Hat, Fedora:** `# yum install git`

**En Mac** Hay dos maneras de instalar Git en Mac, la más fácil es utilizar el instalador gráfico:

[Git for OS X](#)

**En Windows** Para instalar git en Windows debemos descargar el programa instalador en su web oficial en <http://git-scm.com/downloads>.

Una vez descargado, sólo tenemos que ejecutarlo y se abrirá una ventana que nos irá solicitando paso a paso los datos necesarios para la instalación. Pulsaremos el botón “Next” para comenzar.

Nos pasará a la página de licencia (*es una licencia libre que permite copiar, modificar y distribuir el programa*).

La siguiente es una ventana que nos permite elegir el lugar de instalación. Si no tenemos especial interés en que sea otro, el que viene por defecto está bien.



Figure 3: Instalación de git en Windows (1)

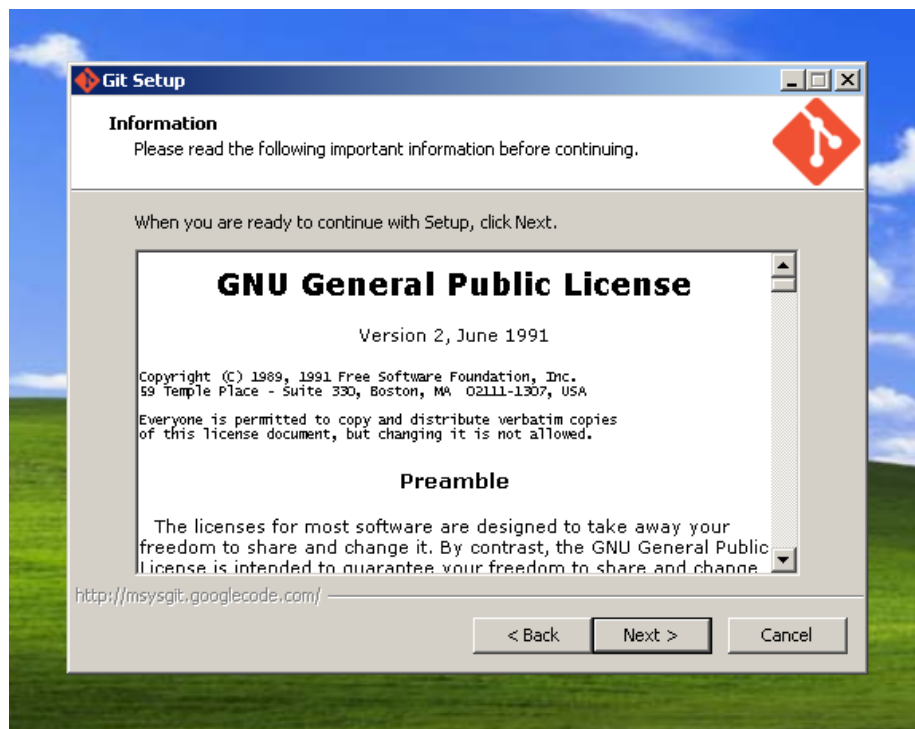


Figure 4: Instalación de git en Windows (2)

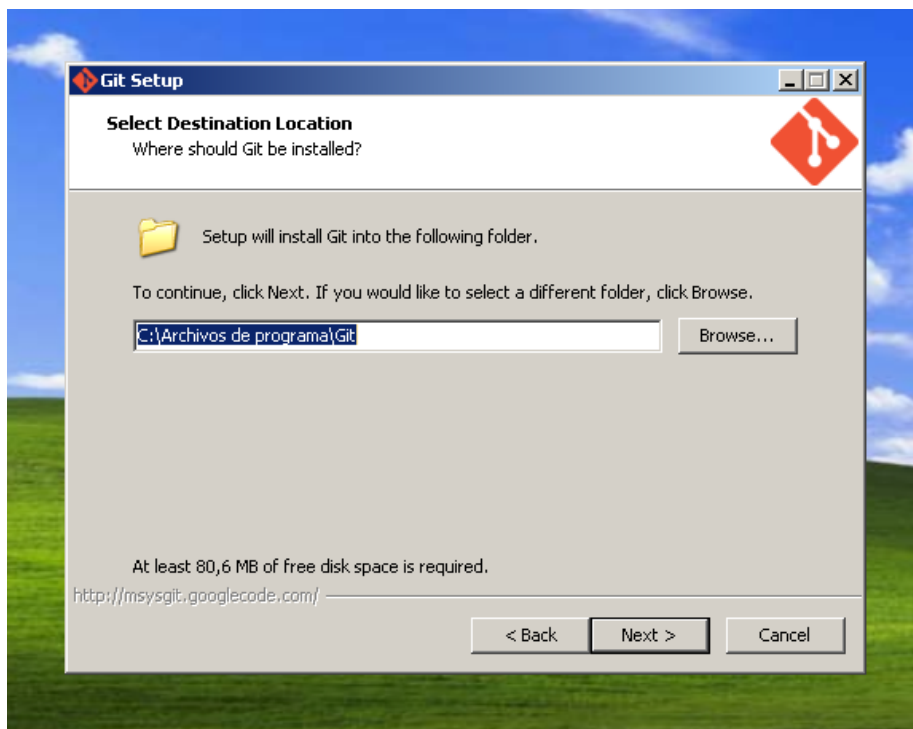


Figure 5: Instalación de git en Windows (3)

En la siguiente, podemos elegir una serie de cosas como el que aparezcan iconos de git en Inicio Rápido y el Escritorio o tener dos nuevas órdenes en el menú contextual (*el que aparece al hacer clic derecho con el ratón en una ventana*) para iniciar una ventana de git en esa carpeta.

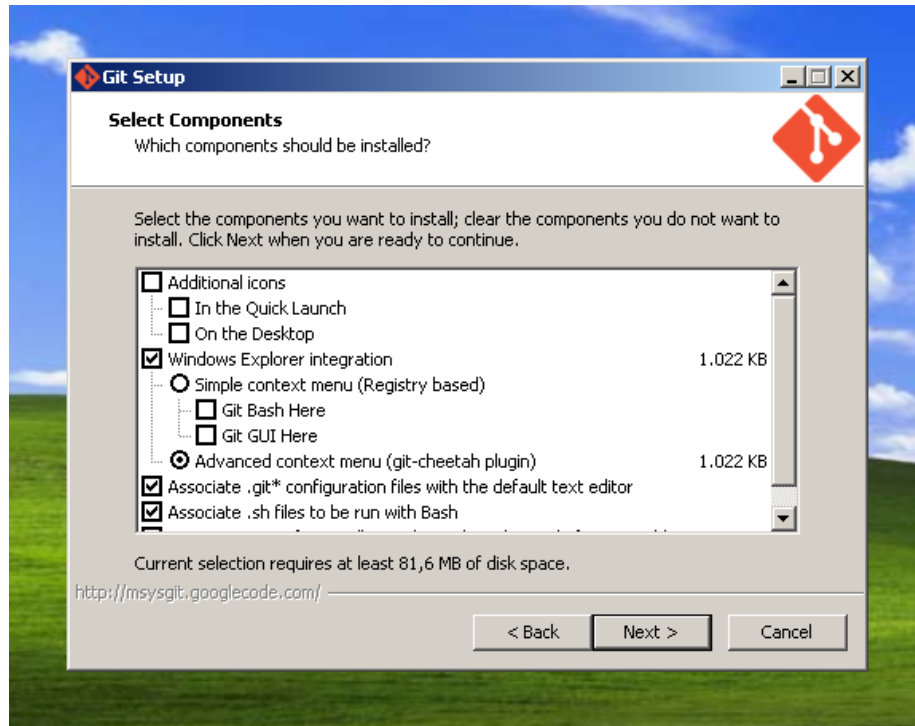


Figure 6: Instalación de git en Windows (4)

En la siguiente ventana se nos permite cambiar el nombre del grupo de programas que aparecerá en el menú Inicio.

Las siguientes dos opciones configuran aspectos avanzados de 'git', concretamente el uso del prompt y el manejo de retornos de carro. Para un usuario novel son adecuadas las opciones por defecto.

Y, por fin, hemos finalizado nuestra instalación.

A partir de este momento podemos ir al menú inicio como se indica en la imagen, y ejecutar "Git Bash", lo que abrirá una consola donde podremos interactuar con 'git' tal como se ve en este curso.

*(Al terminar todos estos pasos, y como se ven en la imagen, también se instalará una versión gráfica "Git GUI", pero en este curso se seguirá la interfaz de línea de comandos)*

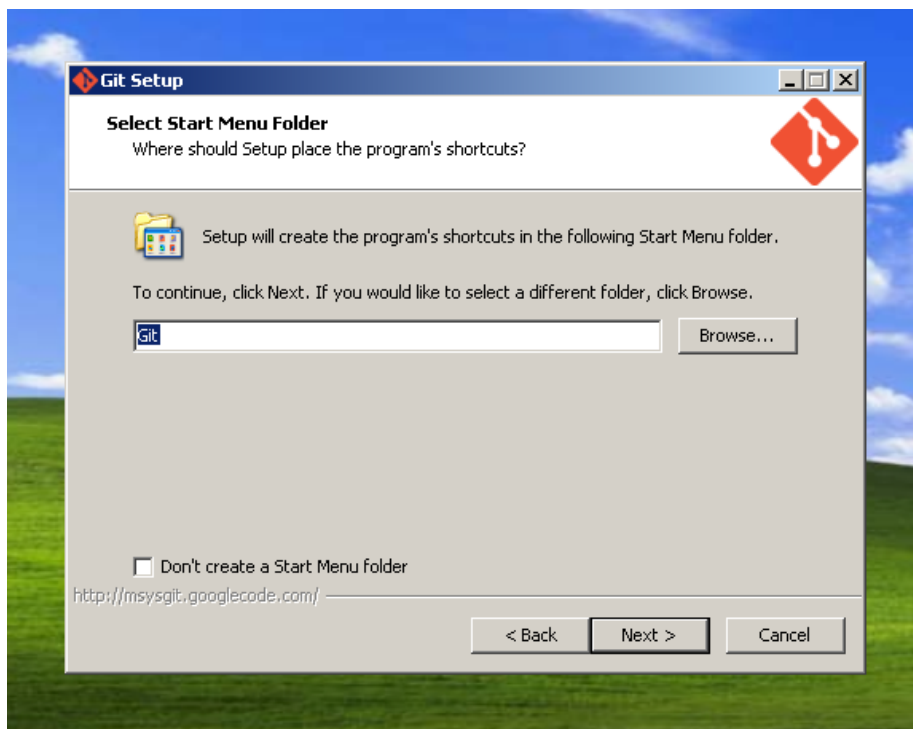


Figure 7: Instalación de git en Windows (5)

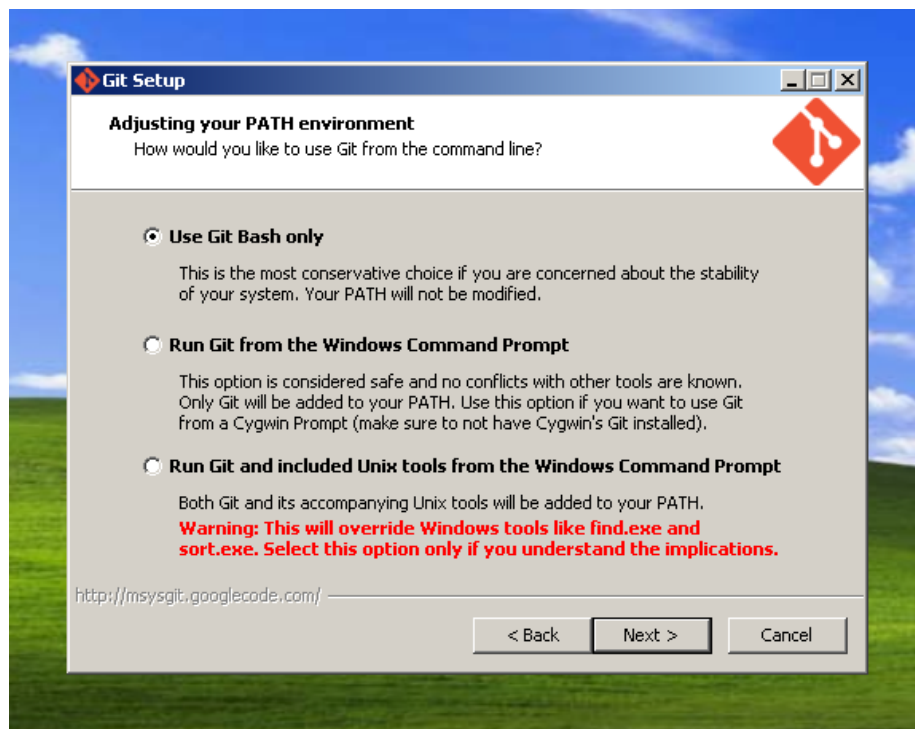


Figure 8: Instalación de git en Windows (6)

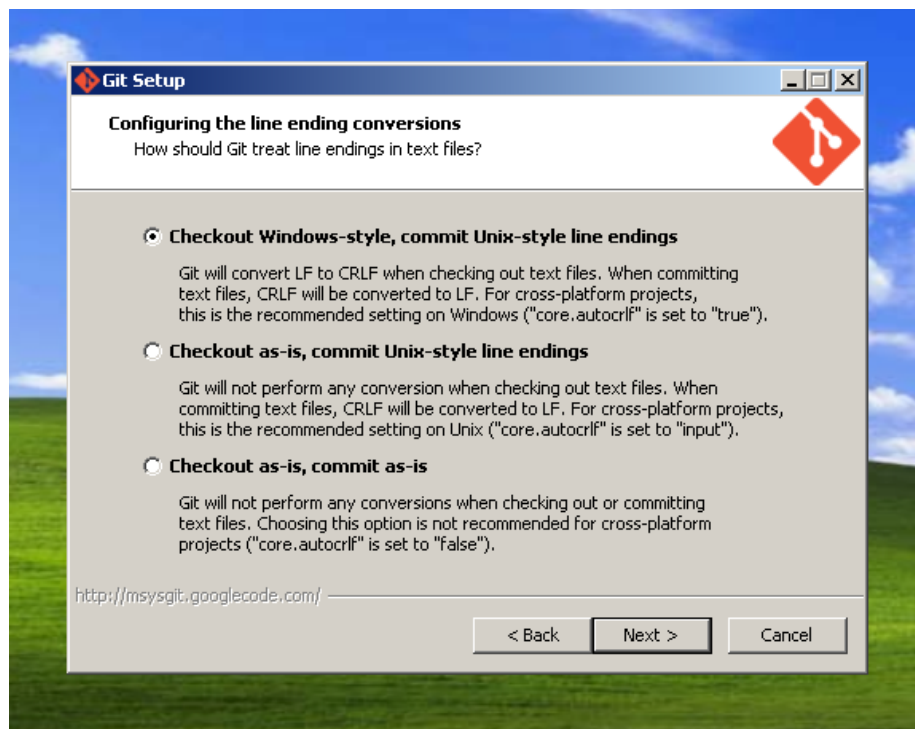


Figure 9: Instalación de git en Windows (7)





Figure 10: Instalación de git en Windows (8)



Figure 11: Instalación de git en Windows (9)

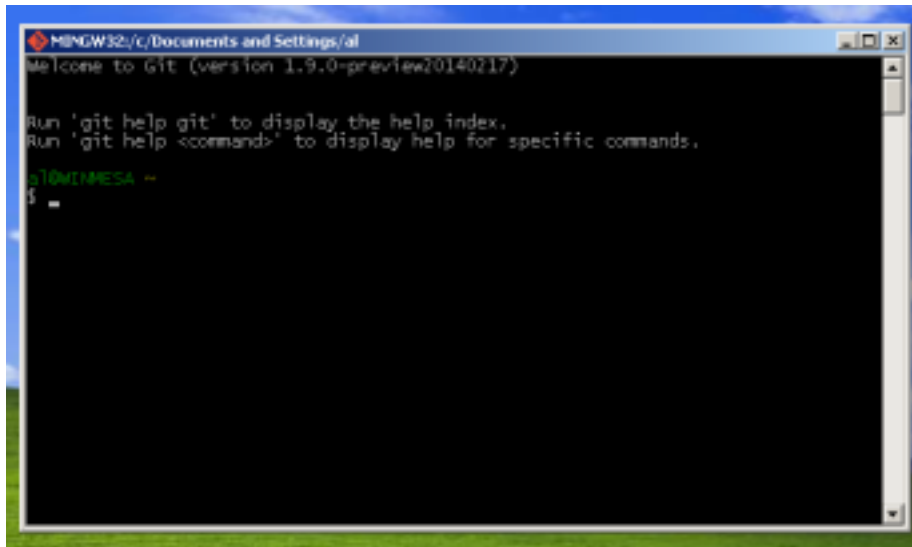


Figure 12: Instalación de git en Windows (10)

Los autores de este libro no somos partidarios del uso de Windows para desarrollo de software. Pero, si tienes que hacerlo porque no te queda otro remedio o porque te gusta, otra alternativa es usar [git for Windows](#) o [msysgit](#), un entorno completo que incluye un intérprete de órdenes y un entorno gráfico para trabajar desde él.

### Clientes GUI para Linux, Windows y Mac

En este curso se seguirá la interfaz de *línea de comandos* (o *línea de órdenes*), pero existen varias aplicaciones para diversos sistemas operativos que permiten interactuar gráficamente (*Interfaz GUI*) con `git` de forma más o menos completa.

[GUI Mac](#)

[GUI Windows](#)

[GUI for Linux, Windows y Mac](#)

### Empezando a usar git

Git es un programa en línea de comandos, y se te supone un conocimiento básico del manejo de esta (cosas como moverse por el árbol de directorios y poco más). No es necesario saber nada complejo, sólo los rudimentos básicos.

**Configurar** Lo primero que hay que hacer antes de empezar a usar git es configurar un par de parámetros básicos que nos identifican como usuario, que son nuestro correo electrónico y nuestro nombre.

Git usará estos datos para identificar nuestros aportes o modificaciones a la hora de mostrarlos en logs etc.

Configurar estos parámetros es muy fácil. Desde la línea de comandos escribimos las siguientes órdenes:

```
git config --global user.name "Nombre que quieras mostrar"
```

y

```
git config --global user.email "correo@electroni.com"
```

¿qué acabamos de hacer? Veámoslo, paso a paso:

Todos los comandos de git empiezan con la palabra **git**.

En este caso, el comando en sí mismo es **config**, que sirve para configurar varias opciones de git, en el primer caso **user.name** y en el segundo **user.email**.

Habrás notado que hay un parámetro **--global** en cada uno de los comandos. Este sirve para decirle a git que esos datos se aplican a todos los repositorios que abras.

Si quieres que algún repositorio concreto use unos datos distintos, puedes llamar al mismo comando, desde el directorio de ese repositorio, pero usando el parámetro **--local** en lugar de **--global**.

Las opciones que configures como **--global** se almacenarán en un archivo de tu carpeta home, llamado **.gitconfig**. Las opciones **--local** lo harán en un archivo **config** dentro del directorio **.git** de tu proyecto.

Hay más opciones que se pueden configurar, puedes verlas (y ver los valores que tienen) con el comando:

```
git config --list
```

Si te has equivocado al escribir alguno de estos datos o quieres cambiarlo, sólo tienes que volver a ejecutar el comando correspondiente de nuevo, y sobrescribirá los datos anteriores.

Una opción de configuración muy cómoda es **git config --global color.ui true**, que hace que el interfaz de git use (si es posible) colores para resaltar distintos aspectos en el texto de sus mensajes.

**Iniciando un repositorio** Un repositorio de git no es más que un directorio de nuestro ordenador que está bajo el control de git. En la práctica, esto significa que en el directorio raíz de nuestro proyecto hay otro directivo oculto llamado “.git” donde se guardan, por ejemplo, los archivos para el control de historiales y los cambios.

Para iniciar un repositorio sólo hay que situarse en el directorio de nuestro proyecto (el que contiene o va a contener los archivos que queremos controlar) y ejecutar la siguiente orden:

```
git init
```

Si todo va bien, este comando responderá algo parecido a “Initialized empty Git repository in /ruta/a/mi/proyecto/.git/”, que significa que ya tienes creado tu primer repositorio. Vacío, pero por algo hay que empezar.

**Clonando un repositorio** Un repositorio también puede iniciarse copiando (*clonando*) otro ya existente.

```
git clone REPOSITORIO
```

por ejemplo:

```
git clone https://github.com/oslugr/repo-ejemplo.git
```

Git usa su propio protocolo “git” para el acceso remoto (también se puede clonar un repositorio local, simplemente indicando el path), pero también soporta otros protocolos como **ssh**, **http**, **https**...

Al contrario que con **git init**, con **git clone** no es necesario crear un directorio para el proyecto. Al clonar se creará un directorio con el nombre del proyecto dentro del que te encuentres al llamar a la orden.

Clonar un repositorio significa copiarlo completamente. No sólo los archivos, sino todo su historial, cambios realizados, etc. Es decir que en tu repositorio local tendrás exactamente lo mismo que había en el repositorio remoto de donde lo has clonado.

Si has clonado el repositorio del ejemplo anterior (y si no, hazlo ahora), podemos ver un par de cosas interesantes. ¿Recuerdas las orden **git config --list**? Entra en el directorio del repositorio (para ello tendrás que hacer algo como **cd repo-ejemplo/**) y lista las opciones de configuración.

Verás, entre otras muchas, las **user.name** y **user.email** que ya conoces. Pero hay otra que es importante, y es **remote.origin.url**, que debe contener la dirección original del repositorio del que has clonado el tuyo.

Ahora mismo no nos sirve de mucho pero, cuando más adelante trabajemos en red con otros repositorios, nos va a venir bien recordarlo.

***IMPORTANTE*** En adelante, a menos que se diga lo contrario, todos los comandos y órdenes que se indique se deberán ejecutar en

el directorio de nuestro proyecto (o uno de sus subdirectorios, lógicamente). Git reconoce el proyecto con el que está trabajando en función del lugar donde te encuentres al ejecutar los comandos

### ¿Cómo funciona git?

Antes de continuar, vamos a detenernos un momento para entender el funcionamiento de git.

Cuando trabajas con git lo haces, evidentemente, en un directorio donde tienes tus archivos, los modificas, los borras, creas nuevos, etc.

Ese directorio es lo que llamamos “Directorio de trabajo”, puede contener otros directorios y, de hecho es el que contiene el directorio `.git` del que hablábamos al principio.

Git sabe que tiene que controlar ese directorio, pero no lo hace hasta que se lo digas expresamente.

Más adelante veremos con algo más de detalle la orden `git add`, pero ya te adelanto que lo que hace es preparar los archivos que le indiques poniéndolos en una especie de lista virtual a la que llamamos el “Index”. En Index ponemos los archivos que hemos ido modificando, pero las cosas que están en el “Index” aun no han sido archivadas por git.

Ojo, que algo esté en el index no significa que se borre de tu directorio de trabajo ni nada parecido, el Index es sólo una lista de cosas que tendrás que actualizar en el repositorio porque han cambiado.

Por último, la instrucción `git commit`, que también veremos en breve, es la que realmente envía las cosas que hay en el Index al repositorio. Solo que, en lugar de “repositorio” lo vamos a llamar “HEAD”, porque el lugar exacto al que va puede significar cosas distintas en según que casos, como ya veremos cuando hablemos de ramas y esas cosas.

Lo sé, es todo un poco lioso ahora mismo, pero ya se irá aclarando conforme aprendamos más cosas.

Tú sólo mantén esta secuencia en la cabeza: Directorio de trabajo -> Index -> HEAD

### Manteniendo nuestro repositorio al día

Tienes tu repositorio iniciado (o clonado) con una serie de archivos con los que empiezas a trabajar, creándolos, editándolos, modificándolos, etc.

Para que git sepa que tiene que empezar a tener en cuenta un archivo (a esto se le llama *preparar* un archivo), usamos la orden `git add` de este modo:

```
git add NOMBRE_DEL_ARCHIVO
```

Esto, como vimos antes, añadirá el archivo indicado con `NOMBRE_DEL_ARCHIVO` al Index. No lo archivará realmente en el sistema de control de versiones ni hará nada. Sólo le informa de que debe tener en cuenta ese archivo para futuras instrucciones (que es, básicamente, en lo que consiste el Index).

Si intentas añadir al Index un archivo que no existe te dará un error.

También puedes usar *comodines*, con cosas como:

```
git add miarchivo.*
```

(que reconocería, por ejemplo “miarchivo.txt”, “miarchivo.cosas” y “mi-archivo.png”)

o

```
git add miarchivo$.txt
```

(que identificaría cosas como “miarchivo1.txt”, “miarchivo2.txt” y “mi-archivoZ.txt”)

Y, en general, todos los comodines que permita usar tu sistema operativo.

Si, en lugar de un archivo, indicas un directorio, se agregarán al Index todos los archivos de ese directorio.

De este modo, la forma más fácil de agregar todos los archivos al Index es mediante la orden:

```
git add .
```

que añadirá el directorio en el que te encuentras y todo su contenido (incluyendo subdirectorios etc).

Un detalle importante es que, si mandas algo al Index con `git add` y luego lo modificas, no tendrás en Index la última versión, sino lo que hayas hecho hasta el momento del hacer el add.

Esto es muy útil (a veces tienes que hacer cambios que aún no quieres “archivar”) pero puede llevarte a alguna confusión.

Ahora vamos a ver una orden que será tu gran amiga:

```
git status
```

`git status` te da un resumen de cómo están las cosas ahora mismo respecto a la versión del repositorio (concretamente, respecto al HEAD). Qué archivos has modificado, que hay en el Index, etc (también te cuenta cosas como en qué rama estás, pero eso lo veremos más adelante). Cada vez que no tengas muy claro que has cambiado y qué no, consulta `git status`.

En principio, si no has modificado nada, el mensaje básico que te da `git status` es este:

```
## On branch master
nothing to commit (working directory clean)
```

Pero, y esa es una cosa que vas a ver a menudo en git, si hay algo que hacer te informa de las posibles acciones que puedes llevar a cabo dependiendo de las circunstancias actuales diciendo como, por ejemplo, (use "git add <file>..." to update what will be committed)".

Cuando ya has hecho los cambios que consideres necesarios y has puesto en el Index todo lo que quieras poner bajo el control de versiones, llega el momento de "hacer commit" (también se le llama "*confirmar*"). Esto significa mandar al HEAD los cambios que tenemos en el Index, y se hace de este modo:

```
git commit NOMBRE_DEL_ARCHIVO
```

Como te estarás imaginando, aquí también puedes usar comodines del mismo modo que vimos en git add. Además, si haces simplemente

```
git commit
```

Esto mandará todos los cambios que tengas en el Index.

AL hacer un commit se abre automáticamente el editor de texto que tengas por defecto en el sistema, para que puedas añadir un comentario a los cambios efectuados. Si no añades este comentario, recibirás un error y el commit no se enviará.

```
Puedes cambiar el editor por otro de tu gusto con git config
--global core.editor EDITOR', por ejemplo:git config --global
core.editor vim'
```

Si no quieres que se abra el editor puedes añadir el comentario en el mismo commit del siguiente modo:

```
git commit -m "Comentario al commit donde describo los cambios"
```

Recuerda lo que dijimos antes: si modificas un archivo después de haber hecho git add, esos cambios no estarán incluidos en tu commit (si quieres incluir la última versión, no tienes más que volver a hacer git add antes del commit).

Ahora nos puede surgir un problema:

Si sólo podemos confirmar con commit de un archivo que hayamos preparado con add, y sólo podemos hacer add de un archivo que existe en nuestro directorio de trabajo ¿Cómo le decimos a git que elimine un archivo del repositorio? Para ello tenemos la orden:

```
git add -u
```

que agregará al Index la información de los archivos que deben ser borrados.

Muy similar a la anterior, git add -A sirve para hacer git add -u (preparar los archivos eliminados) y git add . (preparar todos los archivos modificados) en una sola orden.

Un efecto parecido se puede conseguir con

```
git commit -a
```

Esta orden sirve para confirmar todos los cambios que haya en el directorio de trabajo *aunque no hayan sido preparados* (es decir, aunque no hayas hecho `add`). Esto incluye tanto los ficheros modificados como los eliminados, con lo que sería equivalente a hacer un `git add -A` seguido de un `git commit`.

Esta opción ahorra escribir órdenes, pero también te da más oportunidades de meter la pata. En general se recomienda usar por separado `adds` y `commits`, convenientemente salteados de `git status` para comprobar que todo va bien.

## Sincronizando repositorios

Como sistema de control de versiones distribuido, una de las principales utilidades de `git` es poder mantener distintos repositorios sincronizados (es decir, que contengan la misma información), exportando e importando cambios.

Para importar (o exportar) cambios de un repositorio remoto se necesita, lógicamente, tener acceso de lectura a ese repositorio (En sentido estricto, ya hemos importado el estado de un repositorio cuando lo clonamos al hacer `git clone`).

Para sincronizar con uno o más repositorios remotos, debemos saber qué repositorios remotos son esos. Para ello tenemos `remote`, que se usa así:

```
git remote
```

Y, seguramente, te retornará algo parecido a

```
origin
```

Lo que nos dice que el repositorio es el que le indicamos como “origin” al hacer el `clone` y, la verdad, no es mucha información.

Para obtener algo más útil, prueba a hacerlo con el parámetro `-v` de este modo:

```
git remote -v
```

lo que te retornará algo parecido a esto:

```
origin https://github.com/oslugr/repo-ejemplo.git (fetch)
origin https://github.com/oslugr/repo-ejemplo.git (push)
```

Esto te dice que hay un repositorio llamado “origin” que se usará tanto para recibir (fetch) como para enviar (push) los cambios. “origin” es el nombre del



repositorio remoto por defecto, pero puedes tener muchos más y sincronizar con todos ellos.

Para añadir otro repositorio remoto se hace con la misma instrucción `remote` de este modo:

```
git remote add ALIAS_DEL_REPOSITORIO DIRECCION_DEL_REPOSITORIO
```

Donde `ALIAS_DEL_REPOSITORIO` es un nombre corto para usar en las instrucciones de git (el equivalente al “origin” que hemos visto) y `DIRECCION_DEL_REPOSITORIO` la dirección donde se encuentra. Por ejemplo:

```
git remote add personal git://github.com/psicobyte/repo-ejemplo.git
```

Esto añade un repositorio remoto llamado “personal” con la dirección que se indica.

Si ahora hacemos un `git remote -v`, veremos algo como:

```
mio git://github.com/psicobyte/repo-ejemplo.git (fetch)
mio git://github.com/psicobyte/repo-ejemplo.git (push)
origin https://github.com/oslugr/repo-ejemplo.git (fetch)
origin https://github.com/oslugr/repo-ejemplo.git (push)
```

Para eliminar un repositorio tienes:

```
git remote rm NOMBRE
```

Y para cambiarle el nombre:

```
git remote rename NOMBRE_ANTERIOR NOMBRE_ACTUAL
```

Nota que git no comprueba si realmente existen los repositorios que agregas o si tienes permisos de lectura o escritura en ellos, de forma que el hecho de que estén ahí no significa que vayas a poder usarlos realmente.

**Recibiendo cambios** Ha llegado el momento de importar cambios desde un repositorio remoto. Para ello tenemos `git pull` que se usa así:

```
git pull REPOSITORIO_REMOTO RAMA
```

el `REPOSITORIO_REMOTO` es uno de los nombres de repositorio que hemos visto antes (si no pones ninguno, se supone “origin”). Sobre las ramas se hablará un poco más adelante, pero baste decir que, si no ponemos ninguna, se supone que es la rama “master”)

de este modo, la forma más usual de llamar esta orden es, simplemente:

```
git pull
```

(que significaría lo mismo que `git pull origin master`)

Esta instrucción trae del repositorio remoto indicado (o de “origin” si no indicas nada, como hemos visto), todos los cambios que haya respecto al tuyo (lógicamente, no se molesta en traer los que son iguales).

Si el repositorio del que tratas de importar no existe o no tienes permiso de lectura, te dará un mensaje de error advirtiéndote de ello.

Si hay archivos que tú has modificado pero el otro repositorio no, te quedarás con los tuyos. Cuando se trate de archivos que tú no has cambiado pero que sí son distintos en el remoto, actualizarás los tuyos a este último. Pero, si importas archivos que se han modificado en ambos repositorios ¿qué pasa con las diferencias? ¿Sobrescribirá tus archivos? ¿Perderás los del otro repositorio?

Ahí es donde entra la solución de problemas, y lo veremos dentro de poco.

En realidad, `git pull` es la unión de dos herramientas distintas, que son `git fetch`, que trae los cambios remotos creando una nueva rama, y `git merge`, que une esos cambios con los tuyos. En ocasiones te convendrá más usarlas por separado pero, como aún no hemos visto el manejo de las ramas, dejaremos esto por ahora.

**Enviando cambios** Si con `pull` importamos cambios desde otro repositorio, la instrucción `push` es la que nos permite enviar cambios a un repositorio remoto.

Se usa de un modo bastante parecido:

```
git push REPOSITORIO_REMOTO RAMA
```

Igual que hemos visto con `git pull`, los valores por defecto son “origin” para el repositorio y “master” para la rama, con lo que se puede poner simplemente:

```
git push
```

Lo que enviará nuestros cambios al servidor remoto.

Salvo que algo haya cambiado allí.

Si la versión que hay en el servidor es posterior a la última que sincronizamos (es decir, alguien más ha cambiado algo), git mostrará un error y no nos dejará hacer el push. Antes debemos hacer un pull.

Sólo cuando hayamos hecho el pull (y resuelto los conflictos, si es que hubiera alguno), nos dejará hacer el push y enviar nuestra versión.

Al hacer tu push, git te retornará información de los cambios realizados, número de archivos, etc.

**Contraseñas** Naturalmente, como ya hemos comentado, no puedes hacer push a un repositorio en el que no tengas permiso de escritura. Para eso puede ser que sea un repositorio abierto a todo el que conozca la dirección, pero eso

sería muy raro (e inseguro). Lo usual es que cuentes con un usuario y contraseña que te permitan acceder (normalmente por [ssh](#)) al servidor.

En otros repositorios (más raros), también necesitarás usuario y contraseña para acceder a la lectura y, por tanto, para hacer pull.

En ambos casos, git te solicitará el nombre de usuario y la contraseña cada vez que hagas push. No tiene por qué ser muy a menudo, pero puede ser un engorro.

En muchos sitios puedes ahorrarte ese trabajo usando pares de claves ssh. Básicamente consiste en que tu ordenador y el del repositorio se reconozcan entre ellos y no tengas que andar identificándote.

Las instrucciones para hacer esto en github están en [esta página de ayuda](#)

## Comportamiento por defecto de push

Las versiones anteriores de git tenían un comportamiento por defecto a la hora de hacer push llamado 'matching'.

Este consiste en que, al hacer push, se sincronizan todas las ramas del proyecto con sendas ramas en el servidor con el mismo nombre (ya hablaremos en detalle de las ramas más adelante). Si en el servidor no existe una rama con el nombre de alguna local, se crea automáticamente.

La versión 2 de git cambiará ese comportamiento, que pasará a ser **simple**, lo que significa que se sube sólo la rama que tienes activa en este momento a la rama de la que has hecho el pull, pero te dará un error si el nombre de esa rama es distinto.

Mientras tanto, actualmente, git te avisa (a cada push) de que se va a hacer este cambio y te avisa de que puedes configurar este comportamiento por defecto con un mensaje como este:

```
warning: push.default is unset; its implicit value is changing in
Git 2.0 from 'matching' to 'simple'. To squelch this message
and maintain the current behavior after the default changes, use:
```

```
git config --global push.default matching
```

To squelch this message and adopt the new behavior now, use:

```
git config --global push.default simple
```

When push.default is set to 'matching', git will push local branches to the remote branches that already exist with the same name.

In Git 2.0, Git will default to the more conservative 'simple'

behavior, which only pushes the current branch to the corresponding remote branch that 'git pull' uses to update the current branch.

See 'git help config' and search for 'push.default' for further information. (the 'simple' mode was introduced in Git 1.7.11. Use the similar mode 'current' instead of 'simple' if you sometimes use older versions of Git)

Para elegir el comportamiento que prefieres sólo tienes que usar, como ya hemos visto para otras configuraciones, el comando `git config` de este modo:

```
git config --global push.default OPCION
```

Por ejemplo:

```
git config --global push.default matching
```

Usaría la opción `matching` en todos tus repositorios, pero:

```
git config --local push.default simple
```

Usaría la opción `simple` sólo en el repositorio en el que te encuentras.

Otras opciones posibles son:

- `current`: Sube los cambios de la rama activa a una rama remota del mismo nombre. Si no existe esa rama remota, se crea.
- `nothing`: Esta opción sólo tiene sentido para test, debugs y esas cosas. Al hacer push no se subirá nada a repositorio remoto.
- `upstream`: Al igual que `simple`, sube la rama que tienes activa a la rama de la que has hecho el pull pero, en este caso, *no* te dará error si el nombre de esa rama es distinto.

## El archivo `.gitignore`

Cuando hacemos `git add .` o algo parecido, preparamos todos los archivos que hayan sido modificados. Esto es, sin duda, mucho más cómodo que ir añadiendo los archivos uno a uno. Pero muy a menudo hay montones de archivos en tu directorio de trabajo que no quieres que se añadan nunca. Archivos de contraseñas, temporales, borradores, binarios compilados, archivos de configuración local...

Por ejemplo, muchos editores de texto mantienen una copia temporal de los archivos que estás editando, con el mismo nombre pero terminado en el signo “~”. Si haces `git add .`, estos archivos se acabarán añadiendo a tu repositorio, cosa que no tiene demasiada utilidad.

Para evitar este problema y facilitarte el trabajo, git nos permite crear un archivo (varios, en realidad, como veremos enseguida) donde describir qué archivos quieres ignorar.

El archivo en cuestión debe llamarse “*.gitignore*” (empezando por un punto) y ubicarse en el directorio raíz de tu proyecto.

En este archivo podemos incluir los nombres de archivos que queramos ignorar. Por ejemplo, imaginemos que nuestro *.gitignore* tiene este (poco útil) contenido:

```
## Los archivos que se llamen "passwords.txt" serán ignorados
passwords.txt
```

Las líneas de *.gitignore* que comienzan con el signo “#” son comentarios (útiles para quién lo lea), y git las ignora.

Gracias a esto, git ignorará cualquier archivo que se llame passwords.txt, y no los incluirá en tus adds.

Esto es demasiado simple y no nos va a ser muy útil pero, afortunadamente, *.gitignore* permite comodines y otras herramientas útiles. Por ejemplo:

```
## Ignoramos todos los archivos que terminen en "~"
*~

## Ignoramos todos los archivos que terminen en ".temp"
*.temp

## Ignoramos todos los archivos que se llamen
## "passwords.txt", "passwords.c", "passwords.csv"...
passwords.*
```

El archivo *.gitignore* permite hacer cosas mucho más complejas, aunque para la mayoría de los casos con algo como lo visto arriba es suficiente.

Pese a que cada repositorio puede tener su propio *.gitignore*, puede ser útil tener además un archivo general para todos los repositorios.

Git busca por defecto este archivo general en el directorio “*.config/git/ignore*” de tu directorio “Home”, pero esto puede cambiarse con la siguiente orden:

```
git config --global core.excludesfile RUTA_AL_ARCHIVO_IGNORE
```

Por ejemplo, para usar un archivo llamado “ignorar” en mi directorio personal, pondría algo así:

```
git config --global core.excludesfile ~/ignorar
```

El símbolo “~” en un path significa “El directorio Home del usuario”

Puedes encontrar muchos ejemplos de archivos *.gitignore* en este [repositorio de GitHub](#).

Las opciones que se establecen con `git config` para el repositorio local se almacenan permanentemente en el fichero `.git/config`. Por lo pronto no nos preocupemos de este fichero, pero todas las variables anteriores (y alguna más) se pueden poner directamente en este fichero.

## Solución de problemas con git

### Objetivos

En todo desarrollo colaborativo aparecen problemas. No es difícil solucionarlos.

- Saber obtener información sobre git
- Conocer el estado e historial de nuestros proyectos
- Modificar y recuperar estados anteriores
- Solucionar conflictos entre repositorios

### Obtener Ayuda

Lo primero que necesitamos a la hora de enfrentarnos a las dificultades es conocer nuestras herramientas.

Git dispone de una ayuda detallada que nos resultará muy útil. Para invocarla sólo hay que hacer

```
git help
```

también se puede obtener ayuda de un comando concreto con `git help COMANDO`, por ejemplo:

```
git help commit
```

### Viendo el historial

Has hecho una serie de modificaciones seguidas de commits con sus comentarios ¿Cómo puedes ver todo eso? Para ello tienes la instrucción

```
git log
```

La orden `git log` te mostrará un listado de todos los cambios efectuados, con sus respectivos comentarios, empezando desde el más reciente hasta el más antiguo.

El formato de cada commit en la respuesta es como este:

```
commit c2ac7c356156177a50df5b4870c72ce01a88ae63
Author: psicobyte <psicobyte@gmail.com>
Date: Sun Mar 30 11:54:15 2014 +0200
    Cambios menores en las explicaciones de git add y git status
```

La primera línea es un *hash* único que identifica al commit (y que más adelante no será muy útil), seguida del autor, la fecha en que se hizo y el comentario que acompañó al commit.

Por defecto, `git log` nos mostrará todas las entradas del log. para ver sólo un número determinado sólo tienes que añadirle el parámetro `-NUMERO`, donde NUMERO es el número de entradas que quieres ver, por ejemplo:

```
git log -4
```

mostrará las últimas cuatro entradas en el log.

Otra opción posible, si quieres ver una versión más resumida y compacta de los datos, es `--oneline`, que te muestra una versión compacta.

Si, por el contrario, quieres más detalles, la opción `-p` te mostrará, para cada commit, todos los cambios que se realizaron en los archivos (en formato diff).

Otra ayuda visual es `--graph`, que dibuja (con caracteres ASCII) un árbol indicando las ramas del proyecto (ya veremos eso un poco más adelante).

`git log` tiene un montón de opciones más (para filtrar por autor o fecha, mostrar estadísticas...) que, además, se pueden usar en combinación. Por ejemplo, la instrucción

```
git log --graph --oneline
```

Mostrará los commits en versión compacta y dibujando las ramas (cuando las haya), dando una salida parecida a esta:

```
* 6ad05c1 Sólo una cosilla
* 0678363 Resuelve conflicto como ejemplo
|\
| * bf454ef Prueba para crear conflictos. Así mismo.
* | 8785174 Añade título nueva sección
|/
* afee5ab Acaba un-solo-usuario y listo para conflictos
* fd04eff Corregido (más o menos) el markdown
```

Para más detalles, recuerda que `git help log` es tu amigo.

## Borrado de archivos

En git se pueden borrar archivos con la orden `git rm`.

```
git rm NOMBRE_DEL_FICHERO
```

Funciona como la propia orden del sistema operativo, con la salvedad de que *tambien* borra el archivo del Index, si estuviera allí. Esto lo hace muy útil en ocasiones.

Si necesitas borrar el archivo del Index pero sin borrarlo de directorio de trabajo (porque, por ejemplo, te has arrepentido y no quieres incluirlo en el próximo commit), se puede hacer con la opción `--cached` del siguiente modo:

```
git rm --cached NOMBRE_DEL_FICHERO
```

Otra opción para hacer esto mismo es con `git reset HEAD` que se usa del siguiente modo:

```
git reset HEAD NOMBRE_DEL_ARCHIVO
```

## Rehacer un commit

Puedes rehacer el último commit usando la opción `--amend` de este modo:

```
git commit --amend
```

Si no has modificado nada en tus archivos, esto simplemente te permitirá reescribir el comentario del commit pero, si por ejemplo habías olvidado añadir algo al Index, puedes hacerlo antes del `git commit --amend` y se aplicará en el commit.

## Deshacer cambios en un archivo

Has cambiado un archivo en tu directorio de trabajo, pero te arrepientes y quieres recuperar la versión del HEAD (la del último commit). Nada más fácil que:

```
git checkout -- NOMBRE_DEL_ARCHIVO
```

## Resolviendo conflictos

Normalmente los conflictos suceden cuando dos usuarios han modificado la misma línea, o bien cuando han modificado un fichero binario; por eso los ficheros binarios **no** deben estar en un repositorio. Te aparecerá un conflicto de esta forma cuando vayas a hacer `push`

```
To git@github.com:oslugr/curso-git.git
! [rejected]          master -> master (non-fast-forward)
error: failed to push some refs to 'git@github.com:oslugr/curso-git.git'
consejo: Updates were rejected because the tip of your current branch is behind
```



consejo: its remote counterpart. Merge the remote changes (e.g. 'git pull')  
consejo: before pushing again.  
consejo: See the 'Note about fast-forwards' in 'git push --help' for details.

El error indica que la *punta* de tu rama está detrás de la rama remota (es decir, que hay modificaciones posteriores a tu última sincronización). Rechaza por lo tanto el **push**, pero vamos a hacer **pull** para ver qué es lo que ha fallado

```
remote: Counting objects: 7, done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 4 (delta 3), reused 0 (delta 0)
Unpacking objects: 100% (4/4), done.
De github.com:oslugr/curso-git
   afee5ab..bf454ef  master    -> origin/master
Auto-merging texto/mas-usos.md
CONFLICTO(contenido): conflicto de fusión en texto/mas-usos.md
Automatic merge failed; fix conflicts and then commit the result.
```

El conflicto de fusión te indica que no hay forma de combinar los dos ficheros porque hay cambios en la misma línea. Así que hay que arreglar los conflictos. En general, si se trata de este tipo de conflictos, no es complicado. Al mirar el fichero aparecerá algo así

```
<<<<<< HEAD
## Resolviendo conflictos en `git`
=====
## Vamos a ver cómo se resuelven los problemas en git
>>>>>> bf454eff1b2ea242ea0570389bc75c1ade6b7fa0
```

Lo que uno tiene está entre la primera línea y los signos de igual; lo que hay en la rama remota está a continuación y hasta la cadena que indica el número del commit en el cual está el conflicto. Resolver el conflicto es simplemente borrar las marcas de conflicto (los <<<<< y los >>>>> y los ==) y elegir el código con el que nos quedamos; en este caso, como se ve, es el que aparece efectivamente en este capítulo.

Una vez hecho eso, se puede ya hacer **push** directamente sin ningún problema.

## Retrocediendo al pasado

Para recuperar el estado de tu directorio de trabajo tal como estaba en algún momento del pasado, primero necesitas saber qué momento es ese. Eso se consigue con **git log** que, como vimos, nos devuelve (entre otras cosas) un hash que identifica al commit.

Con ese hash ya podemos hacer un `git reset` del siguiente modo:

```
git reset --hard HASH_DEL_COMMIT_A_RECUPERAR
```

Eso, en cierto modo, volverá atrás en el tiempo, deshará los commits posteriores al indicado y traerá a tu directorio de trabajo los archivos tal y como estaban entonces. Todos los cambios posteriores desaparecerán, así que mucho cuidado.

Como consejo, recuerda hacer `push` antes de jugar con `git reset --hard`. De este modo, si quieres recuperar todo el trabajo posterior, no tienes más que hacer `pull` y los recuperarás de nuevo.

También te puedes salvaguardar usando otra rama para hacer el `git reset --hard` sobre ella, pero el uso de ramas es algo que veremos un poco más adelante.

## Viendo (y recuperando) archivos antiguos

Puedes ver los cambios que hiciste en un commit si haces

```
git show HASH_DE_UN_COMMIT
```

Esto puede ser muy útil, pero aun hay más. Si haces `git show HASH_DE_UN_COMMIT:ruta/a/un/archivo` te mostrará el estado de ese archivo en aquel commit.

Esto nos va a servir para hacer un pequeño truco:

```
git show HASH_DE_UN_COMMIT:ruta/a/un/archivo > archivo_copia
```

La orden anterior nos permite redireccionar la salida de `git show` a un archivo llamado `archivo_copia`, con lo que obtendremos una copia del archivo tal y como estaba en el commit indicado.

## Más usos de git

### Objetivos

- Aprender patrones habituales de flujo de trabajo con `git`
- Aprender a trabajar con las ramas.
- Solucionar los conflictos cuando dos desarrolladores trabajan sobre la misma línea.
- Interpretar la historia en sus diferentes formatos

### Flujos de desarrollo de software (y quizás de otras cosas)

Un sistema como `git` no es independiente de una organización del trabajo. Aunque a priori puedes trabajar como te dé la gana, el que te facilite el uso de

ciertas herramientas hace que sea más fácil usar una serie de prácticas que son habituales en el desarrollo de software (y de otras cosas, como documentación o novelas) para hacer más productivos a los equipos de trabajo y poder predecir con más precisión el desarrollo de los mismos. Por eso, aunque se puede usar cualquier metodología de desarrollo de software con el mismo, `git` funciona mejor con [metodologías ágiles](#) que tienen ciclos más rápidos de producción y de despliegue de nuevas características o de arreglo de las mismas. Las metodologías ágiles son iterativas y en todas las iteraciones están presentes la mayoría de los actores del desarrollo: clientes, desarrolladores, arquitectos; incluso en algunas puede que esté la peña de márketing, a ver si se enteran de lo que está haciendo el resto de la empresa para venderlo (y no al revés, vender cosas que luego obligan al resto de la empresa a desarrollar).

El desarrollo se divide por tanto en estas fases

1. Trabajo con el código. Modificar ficheros, añadir nuevos.
2. Prueba del código. La mayor parte de las metodologías de desarrollo hoy en día, o todas, incluyen una parte de prueba; en casi todos los casos esta prueba está automatizada e incluye test unitarios (que prueban características específicas), de integración y de cualquier otro tipo (calidad de código, existencia y calidad de la documentación).
3. Lanzamiento del producto. Cuando se han incorporado todas las características que se desean, se lanza el producto. El lanzamiento del producto, en el caso de web, incluye un *despliegue* (*deploy*) del mismo, y en el caso de tratarse de otro tipo de aplicación, de un *empaquetamiento*. Se suele hablar, en todo caso, de *despliegue* (aunque sea porque las aplicaciones web son más comunes hoy en día que las aplicaciones de escritorio).
4. Resolución de errores con el código en producción. Si surge algún error, se trata de resolver sobre la marcha (*hotfix*), por supuesto, incorporándolo al código que se va a usar para desarrollos posteriores.

En casi todas estas fases puede intervenir, y de hecho lo hace, un sistema de control de fuentes como `git`; en muchos casos no se trata de órdenes de `git`, sino de funciones a las que se puede acceder directamente desde sitios de gestión como GitHub.

## Organización de un repositorio de `git`

No hay reglas universales para la organización de un repositorio, aunque sí reglas sobre como *no* debe hacerse: todo en un sólo directorio. El repositorio debe estar organizado de forma que cada persona sólo tenga que *ver* los ficheros con los que tenga que trabajar y no se *distriga* con la modificación de ficheros con los cuales, en principio, no tiene nada que ver; también de forma que no se sienta tentado en modificar esos mismos ficheros. Vamos a exponer aquí algunas prácticas comunes, pero en cada caso el sentido común y la práctica habitual de la empresa deberá imponerse...

**Qué poner en el directorio principal** Cuando se crea un repositorio en GitHub te anima a crear un `README.md`. Es importante que lo hagas, porque va a ser lo que se muestre cuando entres a la página principal del proyecto en GitHub y, además, porque te permite explicar, en pocas palabras, de qué va el proyecto, cómo instalarlo, qué prerequisites tiene, la licencia, y todo lo demás necesario para navegar por él.

Otros ficheros que suelen ir en el directorio principal

- `INSTALL` por costumbre, suele contener las instrucciones para instalar. También por convención, hoy en día se suele escribir usando Markdown convirtiéndose, por tanto, en `INSTALL.md`.
- `.gitignore` posiblemente ya conocido, incluye los patrones y ficheros que no se deben considerar como parte del repositorio
- `LICENSE` incluye la licencia. También se crea automáticamente en caso desde Github en caso de que se haya hecho así. No hay que olvidar que también hay que incluir una cabecera en cada fichero que indique a qué paquete pertenece y cuál es la licencia.
- `TODO` es una ventana abierta a la colaboración, así como una lista para recordarnos a nosotros mismos qué tareas tenemos por delante.
- Otros ficheros de configuración, como `.travis.yml` para el sistema de integración continua Travis, `Makefile.PL` o `configure` u otros ficheros necesarios para configurar la librería, y ficheros similares que haga falta ejecutar o ver al instalar la librería. Se aconseja siempre que tengan los nombres que suelen ser habituales en el lenguaje de programación, si no el usuario no sabrá como usarlos.

En general se debe tratar de evitar cargar demasiados ficheros, fuera de esos, en el directorio principal. Siempre que se pueda, se usará un subdirectorio.

**Una estructura habitual con directorio de test** Los fuentes del proyecto deben ir en su propio directorio, que habitualmente se va a llamar `src`. Algunos lenguajes te van a pedir que tengan el nombre de la librería, en cuyo caso se usará el que más convenga. Si no se trata de una aplicación sino de una biblioteca, se usará `lib` en vez de `src`, como en esta [biblioteca llamada NodEO](#) Los tests unitarios irán aparte, en un directorio habitualmente llamado `test`. Finalmente, un directorio llamado `examples` o `apps` o `scripts` o `bin` o `exe` incluirá ejemplos de uso de la biblioteca o diferentes programas que puedan servir para entender mejor la aplicación o para ejecutarla directamente.

**Estructura jerárquica con submódulos** Un repositorio `git` tiene una estructura **plana**, en el sentido que se trata de un solo bloque de ficheros que se trata como tal, a diferencia de otros sistemas de gestión de fuentes centralizados como CVS o Subversion en los que se podía tratar cada subdirectorio como si fuera un proyecto independiente. Pero en algunos casos hace falta trabajar con proyectos en los cuales haya un repositorio que integre el resultado del desarrollo independiente de otros, por ejemplo, una aplicación que se desarrolle conjuntamente con una librería. En ese caso un repositorio `git` se puede dividir en **submódulos**, que son básicamente repositorios independientes pero que están incluidos en una misma estructura de directorios.

Por ejemplo, vamos a incluir el texto de este curso en el repositorio de ejemplo, para poder servirlo como una web también:

```
git submodule add git@github.com:oslugr/curso-git.git curso
Clonar en «curso»...
remote: Reusing existing pack: 14, done.
remote: Counting objects: 4, done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 18 (delta 0), reused 0 (delta 0)
Receiving objects: 100% (18/18), 17.26 KiB, done.
Resolving deltas: 100% (4/4), done.
```

Los submódulos no se clonan directamente al clonar el repositorio. Hay que dar dos comandos: `git submodule init` y `git submodule update` dentro del directorio correspondiente; esta última orden servirá para actualizar submódulos también cada vez que haya un cambio en el repo del que dependan (y queramos actualizar nuestra copia); tras ellos habrá que decir `git pull`, como siempre, para traerse los ficheros físicamente.

De esta forma, el repositorio queda (parcialmente) con esta estructura de directorios:

```
jmerelo@penny:~/txt/docencia/repo-tutoriales/repo-ejemplo$ tree
.
├── curso
│   ├── LICENSE
│   ├── README.md
│   └── texto
│       ├── ganchos.md
│       ├── GitHub.md
│       └── mas-usos.md
```

con el subdirectorio `curso` siendo, en realidad, otro repositorio.

Por ejemplo, podíamos tener una estructura que incluyera subdirectorios para `cliente` (un submódulo) y `servidor` (otro submódulo). Con ambos se puede

trabajar de forma independiente y, de hecho, *residen* en repositorios independientes, pero puede que, en caso de empaquetarlos o desplegarlos de alguna forma determinada (por ejemplo, a un PaaS), queramos hacerlo desde un solo repositorio, como en este caso.

Los submódulos pueden ser un ejemplo de flujos de trabajo: en este caso, hay un flujo desde “las fuentes del manantial” (que puede cambiar de forma independiente su proyecto) hasta “la desembocadura” (nuestro proyecto, que lo usa). Dividir un proyecto en módulos y dejar que personas independientes se encarguen de cada uno, integrándolo todo en un submódulo, por tanto, es una forma simple y sin demasiadas complicaciones de hacerlo.

## Flujos de trabajo con git

Un *flujo de trabajo* es simplemente una forma de organizar las tareas de programación de forma que se conozca, de antemano, qué tareas van detrás de qué tareas y cuál es el destino, en cada momento, del código que se está haciendo. El tener un flujo de trabajo consistente hace que se eviten conflictos y también que el resultado del trabajo sea más predecible; también a evitar problemas y a identificarlos fácilmente.

El flujo de trabajo básico, de un solo usuario, cuando se trabaja con un sistema de control de fuentes y lo hace un solo usuario es el siguiente:

1. `git pull`
2. Trabajo con el código; añadir nuevos ficheros fuentes con `git add`
3. `git commit -a -m "[implícito: este commit] [hace] [Tal cosa]"`  
(o `git -am`, que es lo mismo)
4. `git push`

Fijémonos en el tercer paso, el commit. Primero, conviene hacer siempre `-a`, es decir, `-all` por [varias razones](#):

1. Porque examina todos los ficheros que están siendo seguidos, no sólo los del directorio actual y los que hay por debajo.
2. Porque [hace automáticamente un `git rm` sobre los mismos](#), si es que ha sido borrados.

Lo segundo es decidir cuando se hace el *commit*; lo habitual es que se haga cada vez que se lleve a cabo un cambio significativo, pero ¿qué es un cambio significativo? Pues un cambio más o menos atómico, que incluya todos los ficheros afectados por ese cambio y, lo más importante, que deje, a priori, el código en un estado *sano*. No se debe hacer commit de un código sintácticamente incorrecto, dejarse a medias un cambio o, en general, antes de acabar lo que quiera que uno se propusiera hacer al empezar la sesión de trabajo. No es

imprescindible hacer un **push** por cada commit, pero tampoco estorba hacerlo, sobre todo por si, simultáneamente, alguien está modificando el mismo código.

También conviene escribir **-m** para insertar el mensaje directamente sin que salga el editor de textos (que puede que no sea nuestro editor favorito). En el mensaje podemos escribir cualquier cosa, pero hay que tener en cuenta que lo que escribamos es un predicado cuyo sujeto son los cambios que hemos hecho. Sobre qué hay que escribir hay [muchas versiones](#), pero conviene que el mensaje sea informativo, hable de porqués y de cómo más que de qué (no se puede decir “inserta una función”, eso ya se ve en el código, sino por qué se inserta esa función) y se aconseja también un formato similar al siguiente

Hace tal cosa arreglando el error en el issue #666

Siguiendo la sugerencia de @foodev hemos usado el algoritmo  
Vicentico para resolver ese error y ha quedado monísimo.

Este formato se denomina 50+72 y consiste en usar una primera línea con 50 caracteres (justos, en este caso) que incluya una explicación extendida que esté formateada en líneas de 72 caracteres. Esto es difícil de hacer desde la línea de órdenes, así que tienes dos opciones: usar un *script* que formatee este mensaje, o bien configurar tu editor de forma que use ese formato.

El mensaje de *commit* también admite markdown y de hecho se formateará de esa forma en GitHub (posiblemente también en otros repos) y dentro del mismo admite ciertos atajos como referirse a una tarea o *issue* por el número de la misma o a un desarrollador por el handle. En todo caso, se formatee o no de esa forma, es informativo saber por qué hace lo que hace.

En cuanto al **push** es un evento diferente que el **commit** y hace más cosas, por lo que no debe verse como algo automático tras el mismo. Un **commit** es una unidad mínima de cambio, un **push** envía los contenidos al repo remoto y activa una serie de actividades, como la integración continua, comprobación de código y otra serie de cosas interesantes. Además, hasta que no hacemos **push** no comparamos nuestro código con el que está en **HEAD**, que en ese momento puede estar más adelante o más atrás. Si tienes miedo de provocar un conflicto o de que te lo provoque, no hagas un pull hasta estar seguro de que el código no rompe los tests automáticos y hasta que estés seguro de poder resolver los conflictos que ocurran. ¿Cómo se pueden resolver estos conflictos? Lo veremos a continuación.

Igual que en el caso de los submódulos, no deja de ser simplemente una rutina de trabajo más que un flujo de trabajo si trabaja uno solo. Veremos cómo trabajar en diferentes ramas evitando conflictos.

## Ramas

Las *ramas* son una característica de todos los sistemas de control de fuentes. A todos los efectos, una rama es un proyecto diferente que *surge* de un proyecto principal, aunque nada obliga a que contengan nada en común (por ejemplo, un repositorio puede incluir el código y las páginas web como una rama totalmente diferente). Sin embargo, las ramas, que más bien deberían llamarse *ramificaciones* o *caminos*, son *caminos divergentes* a partir de un tronco común que, eventualmente, pueden combinarse (aunque no es obligatorio) en uno sólo. En la práctica y como [dicen aquí](#) una rama es un nombre para un *commit* específico y todos los commits que son antecesores del mismo.

En **git** las ramas son también parte natural del desarrollo de un proyecto, dado que se trata de un sistema de control de fuentes distribuido. Cada usuario trabaja en su propia rama, que se *fusiona* con la rama principal en el repositorio compartido cuando se hace *push*. Por eso en alguna ocasión puede suceder que, cuando se hace *pull* o incluso *push*, si se encuentra que las ramas que hay en el repo con el que se fusiona y localmente contienen diferente número de *commits*, aparecerá un mensaje que te indicará que se está fusionando con la rama principal; todo esto, incluso aunque no se hayan creado ramas explícitamente. En realidad, *pull* es combinación de dos operaciones: **fetch** y **merge**, como ya se ha visto en [el capítulo de uso básico](#). De hecho [hay quien dice que no debe usarse nunca pull](#).

Por ejemplo, en caso de que se haya borrado un fichero (o, para el caso, hecho cualquier cambio) en un repositorio y se trate de hacer *push* desde el local, habrá un error de este estilo.

```
To git@github.com:oslugr/repo-ejemplo.git
! [rejected]          master -> master (non-fast-forward)
error: failed to push some refs to 'git@github.com:oslugr/repo-ejemplo.git'
consejo: Updates were rejected because the tip of your current branch is behind
consejo: its remote counterpart. Merge the remote changes (e.g. 'git pull')
consejo: before pushing again.
consejo: See the 'Note about fast-forwards' in 'git push --help' for details.
```

En este caso habrá dos ramas, en la *punta* de cada una de las cuales habrá un commit diferente. Se siguen instrucciones, es decir, **git pull**

```
jmerelo@penny:~/txt/docencia/repo-tutoriales/repo-ejemplo$ git pull
remote: Counting objects: 2, done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 2 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (2/2), done.
De github.com:oslugr/repo-ejemplo
61253ec..2fd77db master -> origin/master
```



```

Eliminando Makefile
Merge made by the 'recursive' strategy.
Makefile | 3 ---
1 file changed, 3 deletions(-)
delete mode 100644 Makefile

```

y aparece, efectivamente, el directorio borrado. Habrá que hacer el push de nuevo. Una vez hecho, el repositorio se ha estructurado como se muestra en la imagen:

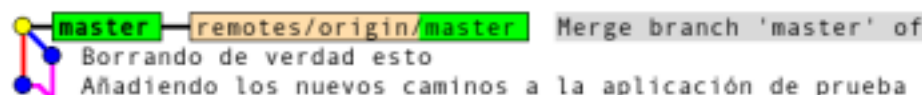


Figure 13: Fusión de dos ramas

Esta imagen, que [se puede ver también en GitHub con fecha 2 de abril](#), y que está obtenida del programa cliente `gitk`, muestra cómo se ha producido la fusión. La que aparece más cerca de la fusión es la que se hizo inicialmente, borrando el fichero, y la más alejada, que aparece más a la izquierda, es la hecha a continuación. El último *commit* fusiona las dos ramas y crea una sola dentro de la rama principal.

Por eso hablamos de enramamiento (bueno, debería ser ramificación, pero esto suena mejor) *natural* en `git`, porque se produce simplemente por que haya dos *commits* divergentes que procedan de la misma rama. Sin embargo, se pueden usar ramificaciones adrede y es lo que veremos a continuación.

**Ramas *ligeras*: etiquetas** Una *etiqueta* permite *guardar* el estado del repositorio en un momento determinado, siendo como una especie de *foto* del estado el proyecto. Se suele asociar a hitos en la historia del mismo: entrada en producción, despliegue de los resultados, o versión mayor o menor.

Para [etiquetar](#) se usa la orden `tag`

```
git tag v0.0.2
```

`tag` etiqueta el último *commit*, es decir, asigna una etiqueta al estado en el que estaba el repositorio tras el último *commit*. La etiqueta aparecerá de forma inmediata (sin necesidad de hacer *push*, puesto que se añade al último *commit* y se puede listar con

```

git tag
jmerelo@penny:~/txt/docencia/repo-tutoriales/repo-ejemplo$ git tag
v0.0.1
v0.0.2

```

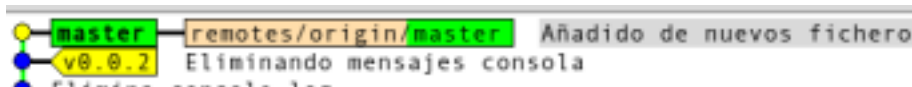


Figure 14: La etiqueta se muestra adosada a un commit

Como se ve, una etiqueta es, en realidad, un apodo para un commit determinado cuyo *hash* puede ser difícil de recordar. Pero por esa misma razón se pueden usar como salvaguarda del estado del repositorio en un momento determinado que, más adelante, se puede recuperar o fusionar con una rama.

Las ramas se pueden también anotar, lo que añade una explicación adicional a lo que ya esté almacenado en el commit correspondiente.

```
git tag -a v0.0.2.1
```

abrirá un editor (el que tengamos especificado por defecto) para añadir una anotación a esta etiqueta, lo que nos podemos ahorrar si usamos

```
git tag -a v0.0.2.1 -m "Estado estacionario del repositorio"
```

por ejemplo. Esta información aparecerá añadida al commit correspondiente (el último que hayamos hecho) cuando hagamos, por ejemplo, `git show v0.0.2.1`

```
tag v0.0.2.1
Tagger: JJ Merelo <jjmerelo@gmail.com>
Date:   Sun Apr 6 09:58:12 2014 +0200
Poco antes de pasar a producción el tema de tags
en realidad, sólo es un ejemplo
commit b958b16b8261fa3ca8159b3ae45e237ae1fa1dce
Author: JJ Merelo <jjmerelo@gmail.com>
Date:   Sun Apr 6 09:45:38 2014 +0200
    Añadido de nuevos ficheros al servidor
    Y edición del README para que sirva para algo
```

(Suprimidos espacios en blanco para que aparezca como un sólo mensaje). Que, como se ve, añade un pequeño mensaje (al principio) al propio del commit (a continuación).

Finalmente, `git describe` es una orden creada precisamente para trabajar con las etiquetas: te indica el camino que va desde la última etiqueta al commit actual o al que se le indique

```
git describe
v0.0.2.1-1-g6dd7a8c
```

que, de una forma un tanto críptica, indica que a partir de la etiqueta v0.0.2.1 hay un commit -1- y el nombre del último objeto, en este caso el único 6dd7a8c. Es otra forma de *etiquetar un punto en la historia de una rama*, o simplemente otra forma de llamar a un commit. Es más descriptivo que simplemente el hash de un commit en el sentido que te indica de qué etiqueta has partido y lo lejos que estás de ella.

Por eso precisamente conviene, como una buena práctica, etiquetar la rama principal con estas *ramas ligeras* cuando suceda un hito importante en el desarrollo. Y también conviene recordar que, dado que son anotaciones locales, *hay que hacer explícitamente git push --tags* para que se comuniquen al repositorio remoto.

**Creando y fusionando ramas** Ya que hemos visto como se crean ramas de forma implícita y de forma *ligera* (con etiquetas), vamos a trabajar explícitamente con ramas. La *forma más rápida de crear una rama* es usar

```
git checkout -b get-dir
Switched to a new branch 'get-dir'
```

Esta orden hace dos cosas: crea la rama, copia todos los ficheros en la rama en la que estemos (que será la **master** si no hemos hecho nada) a la nueva rama y te cambia a la misma; a partir de ese momento estarás modificando ficheros en la nueva rama. Es decir, equivale a dos órdenes

```
git branch get-dir
git checkout get-dir
```

En esta rama se puede hacer lo que se desee: modificar ficheros, borrarlos, añadirlos o hacer algo totalmente diferente.

Un cambio de rama sobrescribirá los cambios que se hayan hecho a los ficheros sin hacer *commit*. Si existen tales cambios te avisará, pero puede que no quieras usar el *commit* para comprometer cambios y dejar el repositorio en un estado incorrecto; en ese caso se puede usar simplemente *git stash* que almacena los cambios en un fichero temporal que se puede recuperar más adelante usando *git stash apply --index*.

En todo momento

```
git status
```

```
# En la rama get-dir
```

nos dirá en qué rama estamos; los ficheros que físicamente encontraremos en el directorio de trabajo serán los correspondientes a esa rama. Conviene hacer siempre `git status` al principio de una sesión para saber dónde se encuentra uno para evitar cambios y sobre todo pulls sobre ramas no deseadas.

La rama que se ha creado sigue siendo rama local. Para crear esa rama en el repositorio remoto y a la vez sincronizar los dos repositorios haremos

```
git push --set-upstream origin get-dir
```

donde `get-dir` es el nombre de la rama que hemos creado. Esta orden establece un origen por defecto (*upstream*) para la rama en la que estamos y además le asigna un nombre a esa rama, `get-dir`.

Las ramas de trabajo se pueden listar con

```
git branch
* get-dir
master
```

con un asterisco diciéndonos en qué rama concreta estamos; si queremos ver todas las que tenemos se usa

```
git branch --all
* get-dir
master
remotes/heroku/master
remotes/origin/HEAD -> origin/master
remotes/origin/get-dir
remotes/origin/master
```

que, una vez más, nos muestra con un asterisco que estamos trabajando en la rama local `get-dir`; a la vez, nos muestra todas las ramas remotas que hay definidas y la relación que hay con las locales, pero más que nada por nombre. Si queremos ver la relación real entre ellas y los commits que hay en cada una

```
jmerelo@penny:~/txt/docencia/repo-tutoriales/repo-ejemplo$ git branch -vv
* get-dir 389b383 [origin/get-dir] Pasado a glob
master 1a93e3d [origin/master] Añade palabros al diccionario
```

con `-vv` indicando doble verbosidad, es decir, que imprima toda la información que tenga.

En este ejemplo se ha mostrado un patrón habitual de uso de las ramas: para probar nuevas características que no sabes si van a funcionar o no y que, cuando

funcionen, se pasan a la rama principal. En este caso se trataba de trabajar con *todos* los ficheros del directorio en vez de los ficheros que le pasemos explícitamente. Estas ramas se suelen denominar *ramas de características o feature branches* y forman parte de un flujo de trabajo habitual en git. Sobre un repositorio central se creará una rama si quieres probar algo que no sabes si estará bien eventualmente o si realmente será útil. De esta forma no se *estorba* a la rama principal, que puede estar desarrollando o arreglando errores por otro lado. En este flujo de trabajo, eventualmente se integra la rama desarrollada en la principal, para lo que se usa pull de nuevo. El concepto de pull es usar primero **fetch** (descargarse los cambios al árbol) y posteriormente **merge** (incorporar los cambios del árbol al índice). En casos complicados esta división te da flexibilidad para escoger qué cambios quieres hacer, pero en un flujo de trabajo como este se puede usar simplemente. Supongamos, por ejemplo, que estamos en la rama **get-dir** y se han hecho cambios en la rama principal.

```
jmerelo@penny:~/txt/docencia/repo-tutoriales/repo-ejemplo$ git pull origin master
De github.com:oslugr/repo-ejemplo
* branch          master      -> FETCH_HEAD
Merge made by the 'recursive' strategy.
 .aspell.es.pws | 3 +++
 README.md      | 5 +++--
2 files changed, 6 insertions(+), 2 deletions(-)
```

Este mensaje te muestra que se ha fusionado usando una estrategia determinada. git examina los commits que diferencian una rama de la otra y te los aplica; al hacer **pull** aparecerá el editor, en el que pondremos el mensaje de fusión. Los cambios se propagarán a la rama remota haciendo **git push** y las ramas quedarán como aparece en [la visualización de la red con fecha 6 de abril de 2014](#); **master** se ha fusionado con **get-dir**.

También podemos hacer la operación inversa. Visto que los cambios de **master** no afectan a la funcionalidad nueva que hemos creado, fusionemos la rama **get-dir** en la principal. Cambiamos primero a ésta

```
git checkout master
```

**git checkout** *saca* del árbol los ficheros correspondientes (lo que puede afectar a los editores y a las fechas de los mismos, que mostrarán la del último checkout si no se han modificado) y nos deposita en la rama principal, desde la cual podemos fusionar, usando también pull

```
jmerelo@penny:~/txt/docencia/repo-tutoriales/repo-ejemplo$ git pull origin get-dir
De github.com:oslugr/repo-ejemplo
* branch          get-dir      -> FETCH_HEAD
Updating df46a37..3705af0
```

```
Fast-forward
package.json | 5 +++--
web.js       | 18 ++++++-----
2 files changed, 15 insertions(+), 8 deletions(-)
```

que, dado que no hemos hecho ningún cambio en el mismo fichero, fusiona sin más problema la rama. En caso de que se hubiera modificado las mismas líneas, es decir, que los *commits* hubieran creado una divergencia, se habría provocado un conflicto que se puede solucionar como se ha visto en el apartado correspondiente. Pero, dado que no la ha habido, el resultado final será el que se muestra en el gráfico.

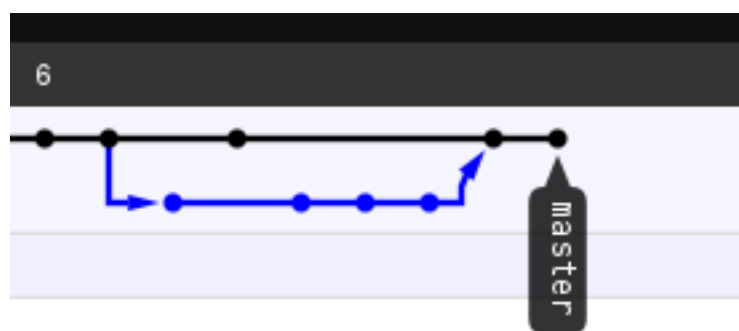


Figure 15: Volviendo al redil del master

La rama, una vez fusionada con el tronco principal, se puede considerar una rama muerta, así que nos la cargamos

```
jmerelo@penny:~/txt/docencia/repo-tutoriales/repo-ejemplo$ git branch -d get-dir
Deleted branch get-dir (was 3705af0).
```

Pero eso borra solamente la rama local. Para [borrarla remotamente](#):

```
jmerelo@penny:~/txt/docencia/repo-tutoriales/repo-ejemplo$ git push origin :get-dir
To git@github.com:oslugr/repo-ejemplo.git
- [deleted]          get-dir
```

Una sintaxis con `:` que es ciertamente poco lógica, pero efectiva. Con eso tenemos la rama borrada tanto local como remotamente.

## Los misterios del rebase

`git` tiene múltiples formas de reescribir la historia, como si de un régimen totalitario se tratara. Una de las más simples es *aplanar* la historia como si todos los *commits* hubieran sucedido unos detrás de otros, en vez de en múltiples ramas como es la forma habitual de trabajar (en un flujo de trabajo *rama por característica* como hemos visto anteriormente). Por ejemplo, podemos crear una rama `img-dir` (sobre el repositorio de ejemplo, añade a la aplicación de forma que se pueda trabajar con las imágenes del tutorial) dejando el repo en el estado que se muestra a continuación.

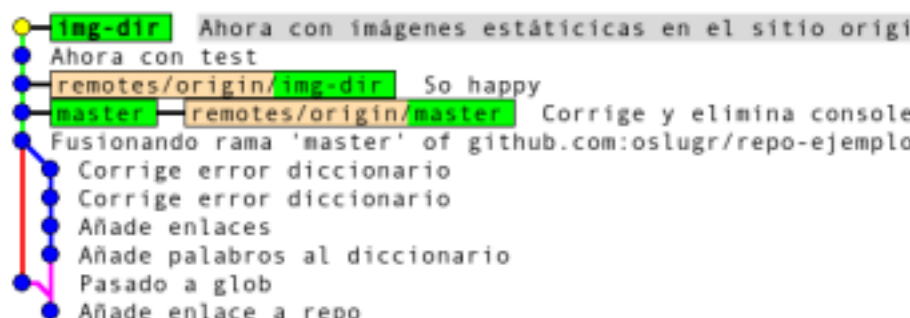


Figure 16: Antes del rebase

tomada desde la propia rama, donde hay una rama `img-dir` con un par de *commits* a partir del máster (dos puntitos azules más abajo).

Una vez acabado el trabajo con la rama, cambiamos a `master` (`git checkout master`) y podemos hacer `rebase`

```
git checkout master
Switched to branch 'master'
jmerelo@penny:~/txt/docencia/repo-tutoriales/repo-ejemplo$ git rebase img-dir
First, rewinding head to replay your work on top of it...
Fast-forwarded master to img-dir.
```

Dejando el repositorio en el estado siguiente

El último commit es ahora parte de la rama `master`. No sólo se han fusionado los cambios en la rama principal, como se ve más abajo en la misma imagen e hicimos con la rama creada anteriormente, `get-dir`. En este caso, y a todos los efectos, se ha *reescrito la historia*, pasando los commits hechos sobre la rama anterior a formar parte de la rama principal. Una vez hecho esto, se limpia eliminando la rama creada. Sin embargo, un rebase no elimina una rama, que sigue ahí, sólo que en una parte diferente del árbol como se muestra a continuación

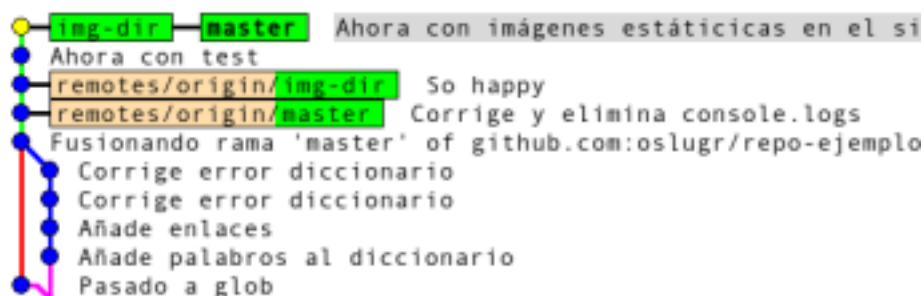


Figure 17: Después del rebase

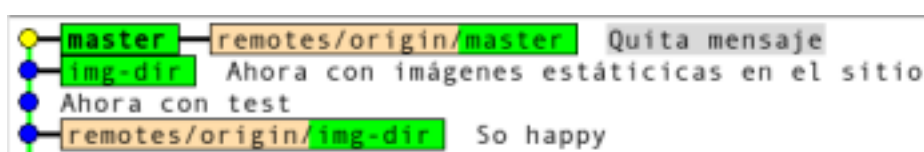


Figure 18: Despegando master de la rama

Sin embargo, ahora la rama es poco menos que un *tag* como el que hemos visto antiguamente. No estorba así que no hace falta borrarla.

### Quién hizo qué

Con todas estas ramificaciones es posible que, en un momento determinado, sea difícil saber quién ha hecho qué cambio. Esto puede ser importante no sólo para repartir las culpas cuando algo falle, sino también para ver quién se responsabiliza de cada rama o característica y, eventualmente, también para asignar méritos. La herramienta `gitk` que hemos usado hasta ahora te presenta en forma de árboles los cambios que se han venido haciendo en el repositorio, con un panel a la derecha que muestra quién ha hecho cada commit:

En esta imagen se ve como cada commit está asignado a uno de los autores de este tutorial, junto con los mensajes correspondientes. Con `git log --pretty=short` se puede conseguir un efecto similar en la línea de órdenes:

```
commit 3b89bd2fffbf7f5988de16b9911b14d70c9197bd
Author: JJ Merelo <jjmerelo@gmail.com>
Añadido texto sobre rebase
commit c09756d4d296fbacd9541d2d7c23e7710a5d1f09
Author: JJ Merelo <jjmerelo@gmail.com>
Añadiendo el capítulo de ramas y tags y esas cosas
commit 8e4559325032fe1425288c4d1ab51fb7072f79b1
Author: psicobyte <psicobyte@gmail.com>
```



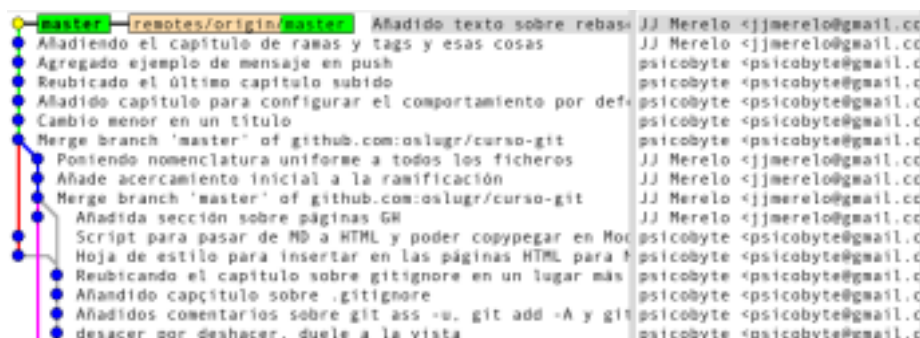


Figure 19: Quien ha hecho qué

#### Agregado ejemplo de mensaje en push

(una vez más, sin líneas en blanco), con una muestra del mensaje corto y del commit junto con el autor. `log` es muy flexible y permite poner cualquier tipo de formato, pero hay todavía más herramientas. `git blame` permite hacer lo mismo sobre un fichero, viendo quién ha modificado cada una de las líneas. Por ejemplo, `git blame uso_basico.md` devolvería, entre otras cosas, estas líneas

```
6017d70c (Manu      2014-03-28 22:30:40 +0100 70) [GUI Mac](http://mac.github.c
2feb1052 (psicobyte 2014-03-10 03:35:49 +0100 71)
6017d70c (Manu      2014-03-28 22:30:40 +0100 72) [GUI Windows](http://windows.
2feb1052 (psicobyte 2014-03-10 03:35:49 +0100 73)
01b9da5a (Manu      2014-03-28 22:36:46 +0100 74) [GUI for Linux, Windows y Mac
75b8e467 (Manu      2014-03-28 22:29:29 +0100 75)
2feb1052 (psicobyte 2014-03-10 03:35:49 +0100 76) ##Empezando a usar git
```

que muestran que la línea 70 y la 71 han sido modificadas por [Manu](#) mientras que el resto lo han sido por [Pablo](#). El formato que siguen es el hash del commit seguidos por el nombre del usuario, la fecha, el número de línea; finalmente está el contenido de la línea. Algo un poco más vistoso se puede ver en algunos repositorios como GitHub, pulsando sobre el botón *Blame* que aparece en cada uno de los ficheros

Esta es simplemente una visualización del comando anterior, que presenta además un enlace al usuario en GH en caso de serlo (porque, recordemos, git es un DVCS cuyos cambios pueden haberse fusionado en local por parte de cualquier tipo de usuario, que no tiene por qué estar necesariamente en GitHub). Con `blame` se puede saber [incluso quien modificó una línea en particular](#). pero, para un uso básico, basta lo anterior.

2Feb1992 = psirebyte 2014-03-18 Segunda mitad, borrador	01	##### MAC
8200000 = MacosX 2014-03-18 update user, build, cd	02	Hay dos maneras de instalar Git en Mac, la más fácil es utilizar el instalador
7900000 = MacosX 2014-03-18 añadir docx 50 y git para Mac	03	
8200000 = MacosX 2014-03-18 GIT MAC	04	[git for OS X](https://code.google.com/p/git-osx-installer/)
2Feb1992 = psirebyte 2014-03-18 Segunda mitad, borrador	05	
	06	##### Git para Linux, Windows y Mac
	07	

Figure 20: Visualización de culpabilidades en GitHub

## Usando git como los profesionales: GitHub

### Objetivos

- Conocer las características generales de los repositorios públicos de GitHub
- Conocer las características específicas de GitHub
- Aprovechar esas características específicas en el trabajo de desarrollo.

### Por qué GitHub

GitHub se ha convertido en el sitio más popular, a pesar de encontrarse entre una cantidad de sitios que alojan también proyectos y permiten usar Git como Gitorious, BitBucket o incluso el venerable [SourceForge](#); [Gitorious](#) tiene la ventaja de que está a su vez basado en software libre, por lo que te puedes instalar tu propia copia del repositorio bajo tu control. Sea con este software o con [Git-Lab](#) te puedes hacer tu propia instalación de git si tienes disponible un servidor para ello. Evidentemente, para trabajar con grandes proyectos privados son una buena opción y lo más aceptable.

Sin embargo, hay buenas razones para usar GitHub. Aparte del uso de git, GitHub en realidad funciona como una comunidad de programadores que te permite interaccionar muy fácilmente con otros programadores que usen las mismas herramientas que uno. Ninguno de los otros repositorios tiene esas características; aunque todos tienen ciertas capacidades *sociales*, lo cierto es que la mayoría de los programadores de software libre usan hoy GitHub. De hecho, GitHub con sus métricas, características de *gamificación* (poner estrellas a proyectos, por ejemplo), continuo desarrollo, enganche a plataformas de programación diversas y cliente móvil es hoy en día la mejor opción para desarrollar software libre (y novelas y [cursos de Git](#)).

Adicionalmente, GitHub se usa como repositorio por defecto para módulos de diferentes lenguajes de programación; por ejemplo, [npm](#), el gestor de módulos de

Node, lo usa para alojar y descargar las fuentes; otros marcos como Joomla lo usan también para definir sus plugins. Incluso GitHub se incluye ya en flujos de trabajo de ciertos entornos, como [la \(futura o pasada\) revista científica Push](#).

En resumen, GitHub es la primera, segunda y tercera mejor opción de un programador de software libre a la hora de alojar sus proyectos. La cuarta sería gitorious, por el hecho de que efectivamente todo el software que usa está liberado y la quinta Bitbucket. Más atrás estarían el resto. En este curso veremos principalmente, por esa razón, GitHub y sus características específicas, algunas de las cuales se han venido usando ya el resto del curso.

## Repositorios públicos y privados

Por omisión, los repositorios de GitHub son públicos. Para conseguir un repositorio privado hay que hacer una de dos cosas

- Pagar. Por 7 dólares al mes puedes tener [cinco repositorios privados](#).
- Conseguir [una cuenta para educación](#) que permite, generalmente de forma limitada, tener un número determinado de cuentas gratuitas.

En realidad, las cuentas privadas son útiles para empresas que quieran desarrollar sus propias aplicaciones en privado; para software libre, aplicaciones que se creen para un proyecto las cuentas tienen espacio de almacenamiento ilimitado y un número ilimitado de colaboradores, por lo que no hace falta adquirir una cuenta ni tener un repositorio privado.

## El GitHub *social*

En realidad, GitHub es una red social de gente que hace cosas y escribe texto como [este](#) o aplicaciones en diferentes lenguajes de programación. Cada usuario tiene [un perfil](#) y en él puede contar cosas como dónde se encuentra, su web y poco más. Lo que importa es lo que se hace, que aparece justo al lado del logo. GitHub usa [Gravatar](#) para las fotos de perfil, por lo que tendrás que darte de alta en ese servicio *con la misma dirección de email que uses en GH* para que aparezca tu avatar junto a tus contribuciones. Se puede seguir la [actividad](#) de un usuario, pero también se puede ir un paso más allá y pulsar el botón de *Follow* con lo que, al entrar en la [página principal de GitHub](#) se te mostrará, junto con la actividad propia, la de esta persona. Una persona puede ser también añadida a un repositorio, lo que le dará privilegios para realizar todo tipo de acciones sobre él. Conviene usar esto con moderación y sólo cuando se trate de una persona ya involucrada en el proyecto.

El componente *social* también se ve en repositorios específicos: los repositorios, como [este de ejemplo del curso](#), se pueden *vigilar* (*Watch*) y también poner una

especie de “Me gusta”, *Star*, uno al lado del otro. Finalmente, se puede hacer un *Fork* del repositorio, lo que copia el contenido completo del repositorio al propio y permite trabajar con él; equivale a una *rama* tal como la que hemos visto anteriormente. Este *fork* respeta la autoría original que sigue apareciendo en todos los *commits* que se hayan hecho originalmente, pero permite añadir uno sus propias modificaciones *sin que el autor original tenga que aprobarlas*. Los *pull requests* permiten colaboración esporádica, ya que las modificaciones que se soliciten pueden aprobarse o no por parte del autor principal del repositorio; la persona que las haga, sin embargo, no tiene por qué estar añadida como colaborador permanente al mismo.

GitHub también permite comentar a diferentes niveles: se puede comentar un *commit*, se pueden comentar líneas de código y finalmente se pueden hacer solicitudes y comentarios a un repo completo usando *issues* o *solicitudes* (*to have an issue* significa literalmente tener un *asunto* o *problema*). Todas estas formas de interacción permiten tener una vida *social* más o menos rica y que haya muchas formas posibles de interaccionar con los autores de un proyecto y por supuesto también de que esos autores aumenten su *karma* a base de las conexiones, estrellas que reciban sus repositorios y comentarios, así como los *issues* resueltos. Los *issues* se agrupan en *milestones* o hitos, que son simplemente grupos de temas que deben ser resueltos antes de pasar a otra fase del desarrollo. Agrupar los *issues* en hitos permite ver el progreso del mismo, ya que te va mostrando cuál es el grado de terminación de dicho hito. Los hitos, además, pueden fecharse con lo que se puede ver si se ha pasado uno de fecha o no.

Finalmente, los usuarios se pueden agrupar en *organizaciones*. Una organización es en muchos aspectos similar a un usuario; tiene las mismas limitaciones y las mismas ventajas, pero en una organización se definen *equipos* y los permisos para trabajar por repositorios se hacen usando estos *equipos*; cada repositorio puede tener uno o más equipos con diferentes niveles de privilegios y el repositorio en sí tendrá también un equipo que será el que pueda realizar ciertas acciones sobre el mismo. Generalmente se crea un equipo por repositorio, pero puede organizarse de cualquier otra forma.

Cuando uno es añadido a un equipo de una organización, se convierte en otra “personalidad” que te permite, por ejemplo, hacer *fork* como miembro de tal organización. Para adoptar esa personalidad se selecciona el nombre de la organización de un menú que está en el panel de control de [GitHub \(página principal\)](#) justo debajo del logotipo del octocat.

En resumen, la facilidad que tiene GitHub para manejar todo tipo de situaciones de desarrollo y la *gamificación* y *socialización* de la experiencia de desarrollo es lo que ha hecho que hoy en día tenga tanto éxito hasta el punto de que el perfil de uno en GitHub es su mejor carta de presentación a la hora de conseguir un trabajo en desarrollo y programación.

## Creando páginas para GitHub pages

Una de las partes interesantes de GitHub y en lo que coincide con otros más clásicos como SourceForge es en la posibilidad de publicar páginas relacionadas con el proyecto o, para el caso, sobre lo que uno quiera. A diferencia de otros sitios, son páginas estáticas (lo que permite, imagino, ser más rápido y eficiente a la hora de servirlos).

También se pueden crear muy fácilmente: *Settings* -> se baja a la página donde pone “GitHub Pages”, y se pulsa en *Automatic Page Generator* que te permitirá elegir entre unos pocos (la verdad, no hay muchos) *temas* el que más te guste.

Lo que hace este generador automático es lo siguiente:

- Generar una rama **gh-pages** de tu repositorio principal
- A partir del fichero **README.md** del directorio principal de tu proyecto, genera un fichero **index.html** usando la plantilla seleccionada
- Genera un dominio **usuario.github.io/proyecto** desde el cual se puede acceder a las páginas publicadas

El generador automático sólo funciona una vez. A partir de ese momento, sólo se reflejarán en el sitio general los cambios que se hagan desde la rama **gh-pages**. Se puede trabajar directamente con ella o bien usar algún tipo de **hook** para generar contenido a partir de la rama **master** y copiarlo a esa rama.

Tampoco es obligatorio usar el generador, que está basado en [Jekyll](#), un motor de plantillas y generador estático de páginas basado en Markdown u otros lenguajes simplificados. Jekyll es muy potente y te permite hasta [montar un blog](#), pero no nos vamos a meter en el funcionamiento del mismo. Tampoco es necesario; para crear una página de proyecto sólo hay que hacer dos pasos:

```
git checkout -b gh-pages
touch index.html
git add index.html
```

(Aquí tendría que editarse el fichero HTML y meter algo, y a continuación)

```
git commit -am "Creada página del sitio"
git push origin gh-pages
```

Con esto se transmite la rama al repositorio y automáticamente se publica.

Adicionalmente a las páginas de proyecto, cada organización y cada usuario puede crear también su página. Un usuario **nombredeusuario** tendrá una página **nombredeusuario.github.io** de la que “colgarán” el resto de las páginas (aunque el realidad se tratará de repositorios diferentes, y sólo estarán las

que se hayan generado, claro). Para crear tanto una página de usuario como de organización simplemente se crea el repositorio y se pone el contenido en la rama principal, la **master**. Al hacer push se publica automáticamente, como la de la [organización que se ha creado para este curso](#)

## Cómo usar los hooks

Los *hooks* o *ganchos* son eventos que se activan cuando se produce algún tipo de acción por parte de git. En general, se usan para integrar el sistema de gestión de fuentes de git con otra serie de sistemas, principalmente de [integración continua](#) o [entrega continua](#) o, en general, cualquier tipo de sistema de notificaciones o de trabajo en grupo.

GitHub puede integrar cualquier tipo de servicio que acepte una petición REST con una serie de características (esencialmente, datos sobre el repositorio y sobre el último commit), pero tiene ya una serie de servicios, casi un centenar, configurables directamente desde el panel de control yendo a *Settings* -> *Webhooks & Services* -> *Configure services*. Todos los servicios se activan cuando se hace un push a GitHub. Evidentemente, en local no se enteran, salvo que los configuremos explícitamente como vamos a ver en el tema siguiente.

Vamos a dividir los servicios que hay en varios grupos:

- Integración continua. Servicios como TravisCI, CircleCI, Jenkins o Shipable. Los tres primeros se pueden configurar directamente desde GH, para el último hay que entrar en [su web](#) y activar el repositorio que haga falta. Estos servicios realizan una serie de tests o generación de código sobre el proyecto y dan un resultado indicando qué tests se han pasado o no. Para indicar qué tests se hacen y los parámetros del repositorio, cada uno usa un formato diferente, aunque son habituales los ficheros de formato YAML o XML. En [el repo de ejemplo](#) se han activado Travis y Shippable, y en el directorio principal se pueden ver los ficheros de configuración (del mismo nombre que el sitio).
- Servicios de mensajería diversos, que envían mensajes cuando sucede algo. Entre estos últimos está [Twitter](#), que se puede configurar para que se cree un tweet con el mensaje del commit cada vez que se haga uno. Puede ser bastante útil, si se usa este sitio, para mantenerte al día de la actividad de un grupo de trabajo. También hay otros servicios como Jabber o Yammer o comerciales como Amazon SNS.
- Entrega continua: a veces integrados con los de, valga la redundancia, integración continua, pero que permiten directamente, cuando se hace un push sobre una rama determinada, se despliegue en el sitio definitivo. Servicios como Azure lo permiten, pero también [CodeShip](#) o [Jenkins](#). Generalmente en este caso hay que configurar algún tipo de *secret* o *clave* que permita a GitHub acceder al sitio y depositar la *carga* que, inmediatamente, estará

disponible. De hecho, es muy fácil trabajar con esto [directamente desde el editor como Eclipse](#)

- Sistemas de trabajo en grupo, que integran GitHub con los sistemas que tengan de asignación de tareas, de resolución de incidencias incluidas por parte de clientes. Por ejemplo, Basecamp, Bugzilla o Zendesk. De hecho, el propio GitHub integra un sistema de incidencias que se puede usar fácilmente.
- Análisis del código como CodeClimate (que analiza una serie de parámetros del código), Depending (que analiza dependencias en PHP) o David-DM (que analiza dependencias para nodejs).

Lo interesante es que se puede trabajar con la mayoría de estos sistemas de forma gratuita, aunque algunos tienen un modelo *freemium* que te cobra a partir de un nivel determinado de uso (lo que es natural, si no no podrían ofrecértelo de forma gratuita). Además, integra la mayor parte de los sistemas que se usan habitualmente en la industria del software.

### **Algunos *hooks* interesantes: sistemas de integración continua**

GitHub resulta ideal para trabajar con cualquier sistema de integración continua, sea alojado o propio. Los sistemas de integración continua funcionan de la forma siguiente:

- Provisionan una máquina virtual con unas características determinadas para ejecutar pruebas o compilar código.
- Instalan el software necesario para llevar a cabo dichas pruebas.
- Ejecutan las pruebas, creando finalmente un informe que indique cuantas han fallado o acertado.
- Crear un *artefacto*, que puede ir desde un fichero con el informe en un formato estándar (suele ser XUnit o JUnit) hasta el ejecutable que se podrá descargar directamente del sitio; esto último puede incluir también su despliegue en la *nube*, un IaaS (Infrastructure as a Service) o PaaS (Platform as a Service) en caso de que haya pasado todos los tests satisfactoriamente.

La integración continua forma parte de una metodología de [desarrollo basado en test o guiado por pruebas](#) que consiste en crear primero las pruebas que tiene que pasar un código antes de, efectivamente, escribir tal código. Las pruebas son tests unitarios y también de integración, que prueban las capas de la aplicación a diferentes niveles (por ejemplo, acceso a datos, procesamiento de los datos, UI). Todos los lenguajes de programación moderno incluyen una aplicación que crea un protocolo para llevar a cabo los test e informar del resultado y estos sistemas

van desde el humilde Makefile que se usa en diferentes lenguajes compilados hasta el complejo Maven, pasando por sistemas como los tests de Perl o los Rakefiles de Ruby. En cualquier caso, cada lenguaje suele tener una forma estándar de pasar los tests (`make test`, `npm test` o `mocha`) y los sistemas de integración continua hacen muy simple trabajar con estos tests estándar, pero también son flexibles en el sentido que se puede adaptar a todo tipo de programa.

Veamos como trabajar con [Travis](#). Se hace siguiendo estos pasos

1. [Darse de alta en Travis CI](#) usando la propia cuenta de GitHub
2. Activar el *hook* en [tu perfil de Travis](#).
3. Se añade el fichero `.travis.yml` a tu repositorio. Este dependerá del lenguaje que se esté usando, aunque si lo único que quieres es comprobar la ortografía de tus documentos, lo puedes hacer [como en el repositorio ejemplo](#)
4. Hacer push.

La mayoría de estos repositorios suelen usar un fichero en formato estándar, YAML, XML o JSON. Veamos qué hace el fichero que hemos usado para el repo ejemplo:

```
branches:
  except:
    - gh-pages
language: C
compiler:
  - gcc
before_install:
  - sudo apt-get install aspell-es
script: OUTPUT=`cat README.md | aspell list -d es -p ./aspell.es.pws`; if [ -n "$OUTPUT" ] ;
```

La estructura de YAML permite expresar vectores y matrices asociativas fácilmente. En general, nos vamos a encontrar con algo del tipo **variable: valor** que será una clave y el valor correspondiente; el valor, a su vez, puede incluir otras estructuras similares. Por ejemplo, la primera

```
branches:
  except:
    - gh-pages
```

es una clave, **branches**, que incluye otra clave, **except**, que a su vez apunta a un vector (diferentes valores precedidos por -) con las ramas que vamos a excluir. En este caso, **gh-pages**, la de las páginas. Si hubiera otra rama a excluir, iría de esta forma



```
branches:
  except:
    - gh-pages
    - a-excluir
```

es decir, como un array de dos componentes. Esto hará que, sobre nuestro repositorio, se prueben todas las ramas, inclusive por supuesto la máster. A continuación expresamos el lenguaje que se va a usar; Travis no admite cualquier lenguaje, pero algunos como C, Perl o nodejs los acepta sin problemas. Como hay varios compiladores posibles, a continuación le decimos qué compilador se va a usar (en realidad, no se usa ninguno). En otros lenguajes habría que decir qué intérprete o qué versión; estas claves son específicas del lenguaje.

```
before_install:
  - sudo apt-get install aspell-es
script: OUTPUT=`cat README.md | aspell list -d es -p ./aspell.es.pws`; if [ -n "$OUTPUT" ] ;
```

Como lo único que vamos a hacer en este caso es comprobar la ortografía del texto del fichero `README.md`, instalamos con `apt-get` (herramienta estándar para Linux) un diccionario en español; este instalará todas las dependencias a su vez. Finalmente, la orden marcada `script` es la que lleva a cabo la comprobación. Para un programa normal sería suficiente hacer `make test` (y definir las dependencias para este objetivo, claro). No nos preocupemos mucho por lo que es, sino por lo que hace: si hay alguna palabra que no pase el test ortográfico, [fallará y enviará un mensaje de correo electrónico a la persona que haya hecho un commit indicándolo](#). Si lo pasa sin problemas, [también enviará el mensaje indicando que todo está correcto](#). Este tipo de cosas resulta útil sólo por el hecho de que se ejecuten automáticamente, pero pueden servir también para hacer despliegues continuos.

Travis también proporciona un *badge* que puedes incluir en tu repositorio para indicar si pasa los tests o no, que puedes incluir en tu fichero `README.md`(o donde quieras) con este código

```
[![Build Status](https://travis-ci.org/oslugr/repo-ejemplo.svg?branch=master)](https://tr
```

sustituyendo el nombre de usuario y el nombre del repo por el correspondiente, claro. Este código está escrito en Markdown, y GitHub lo interpretará directamente sin problemas, aunque lo mejor es que pinches en la imagen que aparece arriba a la derecha que te dará el código correspondiente.

## Cliente de GitHub

GitHub también mantiene un [cliente de GitHub](#), escrito en Ruby y llamado `hub`, que se puede usar para sustituir a `git` o por sí mismo. En realidad, es como `git`

salvo que tiene ya definidos por omisión una serie de características específicas de GitHub, como los nombres de los repositorios o los usuarios de los mismos. Tras [instalarlo](#) puedes usarlo, por ejemplo, para clonar el repo de ejemplo usado aquí con `hub clone oslugr/repo-ejemplo` en vez de usar el camino completo a git; el formato sería siempre `usuario/nombre-del-repo`. Más órdenes que añade a git (y que se pueden usar directamente desde git si se usa, como se indica en las instrucciones, git como un alias de `hub`):

- `hub browse`: abre un navegador en la página del repositorio correspondiente. Por ejemplo `hub browse -- issues` lo abriría en la página correspondiente a las solicitudes de ese proyecto.
- `hub fork`: una vez clonado un repositorio de otro usuario, no hace falta hacer *fork* desde la web, se puede hacer directamente desde el repo. Se crea un origen remoto con el nombre de tu usuario, al que se puede hacer *push* de la forma normal. Desde la misma línea de órdenes se puede hacer un *pull request* al repositorio original también.
- Como usuario puedes aplicar también los *pull requests* desde línea de órdenes.
- `hub compare` permite comparar entre diferentes tags o versiones o ramas.

En general, ya que se tiene GitHub, conviene usar este cliente, sea o no con un alias a git. Por lo menos su uso es conveniente.

### Listo para el lanzamiento: publicación en GitHub

GitHub, como cualquier otro repositorio git, permite usar una rama específica para depositar las versiones descargables; una forma de hacerlo, por ejemplo, es usar la rama `gh-pages` para no mezclarlo con el resto de los ficheros que están en el repositorio y, por tanto, versionados. Sin embargo, no es una buena idea poner ficheros binarios bajo control de git, porque es muy fácil provocar conflictos con ellos y no tan fácil resolverlos (o es un fichero o es otro, los algoritmos de diferencias de texto no trabajan sobre ficheros que no sean binarios); además, en general, estos ficheros se generan a partir del fuente de una forma más o menos automática: usando `Makefiles`, por ejemplo, en C, o en general compilando; si se trata de paquetes, cada lenguaje tiene mecanismos específicos para crearlos a partir de los fuentes, por lo tanto no es necesario colocar tales ficheros en el repositorio. Por eso es mejor colocarlos *fuera* del repositorio, y es lo que hace GitHub, en un apartado llamado *archivos*.

Crear un lanzamiento es fácil en GitHub: simplemente se crea una etiqueta como se ha visto anteriormente, con `git tag`. Por ejemplo, [este es el fichero correspondiente a la etiqueta v0.0.1 que se definió en el repositorio de ejemplo](#). Al pinchar en [Releases](#) te aparecen las versiones que ya has creado o un botón con *Draft a new release* para crear una nueva.

Desde esta interfaz web se puede añadir alguna información más que desde la línea de comandos: se puede crear la etiqueta si no existe y se pueden añadir imágenes, ficheros binarios generados de cualquier otra forma (o automáticamente) y, en general, lo que uno desee. También se puede marcar como *pre-release* y darle un título como a las versiones de Ubuntu, con animalitos o nombres de días de la semana o lo que sea.

En general, si no se usa ningún repositorio de módulos o aplicaciones para publicar la aplicación, o simplemente se quiere publicar junto con los fuentes, manuales y lo que se desee, es conveniente usar esta característica de GitHub para mantener un archivo de versiones descargables de la misma y también para que puedan acceder a ella fácilmente quienes no quieran usar simplemente `git` para descargársela.

Vais a decir que ya podían instalarse `git` y demás herramientas necesarias para compilar o ejecutar la aplicación, pero en muchos casos no tiene por qué ser fácil o factible; no se va a instalar uno un compilador de fortran simplemente para compilar una aplicación nuestra, por ejemplo.

## ***Hooks: ejecutando código tras una orden git***

### **Objetivos**

- Entender el concepto de *fontanería* y *loza*
- Entender el concepto de *hooks* o *puntos de enganche*
- Entender las órdenes menos usuales de git usadas desde los *hooks*
- Saber adaptar *hooks* para una labor determinada

### **Viendo las cañerías: estructura de un repositorio git**

Cuando se crea por primera vez un repositorio veremos que aparecen misteriosamente una serie de ficheros con esta estructura dentro del directorio `.git`.

`branches` lo dejamos de lado porque ya no se usa (aunque por alguna razón se sigue creando). `config`, `HEAD`, `refs` y `objects` son ficheros o directorios que almacenan información dinámica, por ejemplo `config` almacena las variables de configuración (que se han visto anteriormente). El resto de los ficheros y directorios se copian de una *plantilla*; esta plantilla se instala con `git` y se usa cada vez que hacemos `clone` o `init`, y contiene `hooks`, `description`, `branches` e `info` y los ficheros que se encuentran dentro de ellos.

Esta plantilla la podemos modificar y cambiar. La plantilla que se usa por omisión se encuentra en `/usr/share/git-core/templates/` y contiene una serie de ficheros junto con ejemplos (*samples*) para ganchos. Sin embargo, podemos personalizar nuestra plantilla editando (con permiso de superusuario) estos

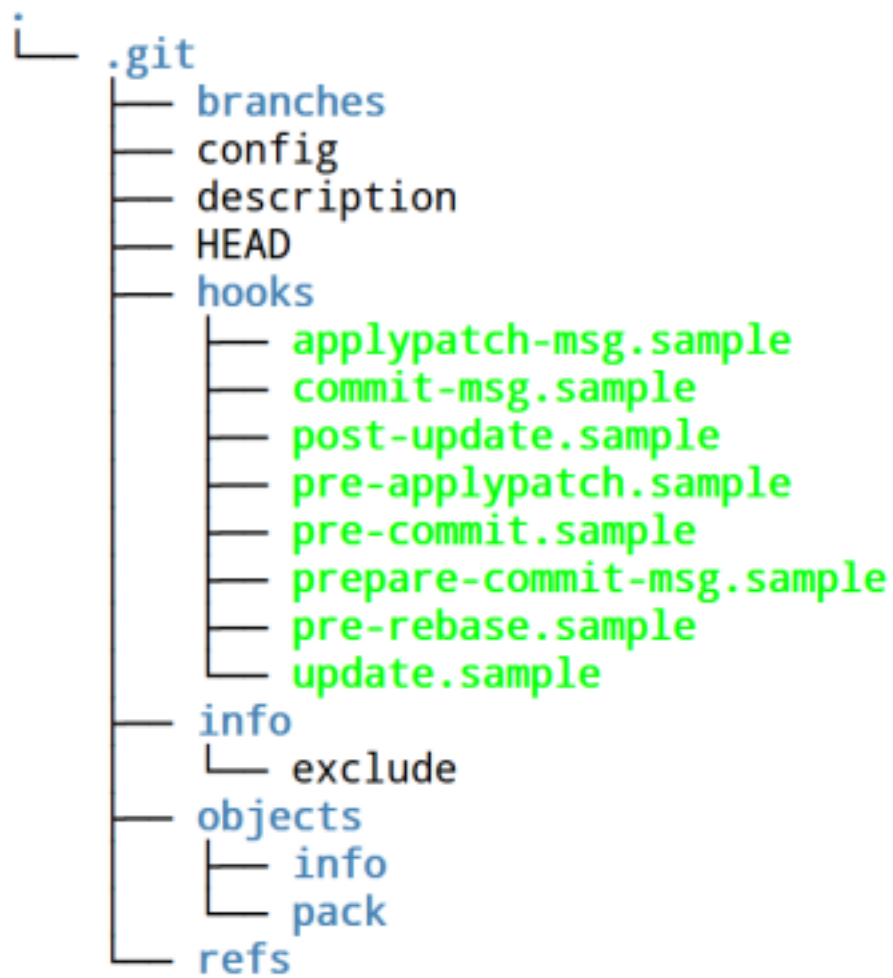


Figure 21: Estructura básica de un repositorio git

ficheros o bien usando la opción `--template <nombre de directorio>` de `clone` o `init`. En ese caso, en vez de copiar los ficheros por omisión, copiará los contenidos en ese directorio.

Por ejemplo, se puede usar [esta plantilla](#) que elimina los ficheros de ejemplo, sustituye por otro y traduce los contenidos de los otros ficheros al castellano; también mete en los patrones ignorados (sin necesidad de usar `.gitignore`) los ficheros que terminan en `~`, que produce Emacs como copia de seguridad.

Estos ficheros forman parte de las cañerías de `git` y podemos cambiar su comportamiento editando `config` como ya se ha visto en el capítulo de uso básico; de hecho, existe también un fichero de configuración a nivel global, `.gitconfig` que sigue el mismo formato y que ya hemos visto

```
[alias]
    ci = commit
    st = status
[core]
    editor = emacs
```

Estos ficheros de configuración siguen un formato similar al de los ficheros `.ini`, es decir, bloques definidos entre corchetes y variables con valor, dentro de ese bloque, a las que se le asigna usando `=`. En este caso [definimos dos alias](#) y un editor o, mejor dicho, *el* editor. Esto podemos hacerlo tanto en el fichero global como en el local si queremos que afecte sólo a nuestro repositorio.

Otro fichero dentro de este directorio que se puede modificar es `.git/info/exclude`; es similar a `.gitignore`, salvo que en este caso afectará solamente a nuestra copia local del repositorio y no a todas las copias del mismo. Por ejemplo, podemos editarlo de esta forma

```
# git ls-files --others --exclude-from=.git/info/exclude
# Lines that start with '#' are comments.
# For a project mostly in C, the following would be a good set of
# exclude patterns (uncomment them if you want to use them):
# *.o[ad]
*~
```

para excluir los ficheros de copia de seguridad de Emacs (que hemos definido antes como editor) que nos interesa evitar a nosotros, pero que puede que tengan un significado especial para otro usuario del repo y que por tanto no quiera evitar.

Por supuesto, el tema principal de este capítulo está en el otro directorio, *hooks*, cuyo contenido tendremos que cambiar si queremos añadir ganchos al repositorio. Pero para usarlo necesitamos también conocer algunos conceptos más de `git`, empezando por cómo se accede a más cañerías.

## Paso a paso

Si miramos en el directorio `.git/objects` encontraremos una serie de directorios con nombres de dos letras, dentro de los cuales están los objetos de git, que tienen otras 38 letras para componer las 40 letras que componen el nombre único de cada objeto; este nombre se genera a partir del contenido usando [SHA1](#).

git almacena toda la información en ese directorio y de hecho está [organizado para acceder a la información almacenada por contenido](#). Por eso tenemos que imaginarlo como un sistema de ficheros normal, con una raíz (que es HEAD, el punto en el que se encuentra el repositorio en este momento) y una serie de ramas que apuntan a ficheros y a diferentes versiones de los mismos.

git, entonces, [procede de la forma siguiente](#).

1. Crea un SHA1 a partir del contenido del fichero cambiado o añadido. Este fichero se almacena en la zona temporal en forma de *blob*.
2. El nombre del fichero se almacena en un árbol, junto con el nombre de otros ficheros que estén, en ese momento, en la zona temporal. Un árbol almacena los nombres de varios ficheros y apunta al contenido de los mismos, almacenado como *blob*. De este objeto, que tiene un formato fijo, se calcula también el SHA1 y se almacena en `.git/objects`. Un árbol, a su vez, puede apuntar a otros árboles creados de la misma forma.
3. Cuando se hace *commit*, se crea un tercer tipo de objeto con ese nombre. Un *commit* contiene enlaces a un árbol (el de más alto nivel) y metadatos adicionales: quién lo ha hecho, cuándo y por supuesto el mensaje de commit.

Veremos más adelante cómo se listan ficheros de todos estos tipos, pero por lo pronto la idea es que un comando de git de alto nivel involucra varias órdenes de bajo nivel que, eventualmente, van a parar a información que se almacena en un directorio determinado con nombres de fichero que se calculan usando SHA1, aparte de que se puede actuar a diferentes niveles, desde el más bajo de almacenar un objeto directamente en un árbol o crear un commit “a mano” hasta el más alto (que es el que estamos acostumbrados). Este sistema, además, asegura que no se pierda ninguna información y que podamos acceder al contenido de un fichero determinado hecho en un momento determinado de forma fácil y eficiente. Pero para poder hacerlo debe haber una forma única y también compacta de referirse a un elemento determinado dentro de ese repositorio. Es lo que explicaremos a continuación.

## El nombre de las cosas: refiriéndonos a objetos en git.

Como ya hemos visto antes, todos los objetos (sean *blobs*, árboles o *commits*) están representados por un SHA1. Si conocemos el SHA1, se puede usar `show`,

por ejemplo, para visualizarlo. Haciendo `git log` veremos, por ejemplo, los últimos commits y si hacemos `show` sobre uno de ellos,

```
git show fe88e5eefff7f3b7ea95be510c6dcb87054bbcb
commit fe88e5eefff7f3b7ea95be510c6dcb87054bbcb0
Author: JJ Merelo <jjmerelo@gmail.com>
Date: Thu Apr 17 18:29:11 2014 +0200
    Añade layout
diff --git a/views/layout.jade b/views/layout.jade
new file mode 100644
index 0000000..36cc059
--- /dev/null
+++ b/views/layout.jade
@@ -0,0 +1,6 @@
[....]
```

El mismo resultado que obtendríamos si hacemos `git show HEAD`, que recordemos que es una referencia que apunta al último commit. También obtendremos lo mismo si hacemos `git show master`. En cualquiera de los casos, lo que está mostrando es un objeto de tipo *commit*, el último realizado.

Pero veremos como funciona este último ejemplo. Al lado del directorio `objects` está el directorio `refs`, que almacena referencias y que es como `git` sabe a qué commit corresponde cada cosa. Este comando:

```
~/repos-git/repo-ejemplo<master>$ tree .git/refs/ .git/refs/
heads      img-dir      master  remotes  heroku
master     origin      HEAD    img-dir  master  tags
v0.0.1     v0.0.2     v0.0.2.1 v0.0.3 5 directories, 10 files
```

muestra todo lo que hay almacenado en este directorio: referencia a las ramas locales en `heads` y a las remotas en `remotes`. Si mostramos el contenido de los ficheros:

```
~/repos-git/repo-ejemplo<master>$ cat .git/refs/heads/master
fe88e5eefff7f3b7ea95be510c6dcb87054bbcb0
```

Que muestra que, efectivamente, el hash del commit es el que corresponde

Podemos mirar en `.git/objects/fe` a ver si efectivamente se encuentra; puedes hacerlo sobre tu copia del repositorio `repo-ejemplo`, ya que los hash son iguales en todos lados.

Como hemos visto anteriormente, un *commit* apunta a un árbol. Podemos indicarle a `show` que nos muestre este árbol de esta forma:

```
~/repos-git/repo-ejemplo<master>$ git show master^{tree}
tree master^{tree}
.aspell.es.pws
.gitignore
.gitmodules
.travis.yml
LICENSE
Procfile
README.md
curso
package.json
shippable.yml
test/
views/
web.js
```

En este caso el [formato es rama \(circunflejo o caret\) {tree}](#); el circunflejo se usa en la selección de referencias de `git` para cualificar lo que se encuentra antes de ella, pero no hay muchas más opciones aparte de `tree`, pero sí podemos acceder a versiones anteriores del repositorio y a sus ficheros.

```
~/repos-git/repo-ejemplo<master>$ git show master~1
commit 5be23bb2a610260da013fcea807be872a4bd6981
Author: JJ Merelo <jjmerelo@gmail.com>
Date: Thu Apr 17 17:42:39 2014 +0200
    Aclara una frase
[...]
```

La [tilde ~](#) indica un ancestro, es decir, el *padre* del commit anterior, que, como vemos [corresponde al commit 5be23bb](#).

La lista completa de opciones para especificar revisiones está, curiosamente, en [la página de referencia del comando rev-parse](#). Hay un número excesivo de ellas, pero si en algún momento no se entiende qué es lo que se está usando, conviene ir ahí.

Podemos ir más allá hasta que nos aburramos: `~2` accederá al padre de este y así sucesivamente. Y, por supuesto, podemos cualificarlo con `^{tree}` para que nos muestre el árbol en el estado que estaba en ese commit. Y también para que nos muestre un fichero sin necesidad de sacarlo del repositorio:

```
~/repos-git/repo-ejemplo<master>$ git show master~2:README.md
repo-ejemplo
=====
```



Ejemplo de repositorio para trabajar en el  
[curso de `git`](http://cevug.ugr.es/git) el contenido del cual está  
[...]

En esta sección hemos usado `show` para mostrar las capacidades de los diferentes selectores de `git`, pero se pueden usar con cualquier otra orden, como `checkout` o cualquiera que admita, en general, una referencia a un objeto.

## Comandos de alto y bajo nivel: *fontanería y loza*

Para entendernos, todas las órdenes que hemos usado hasta ahora son *loza*. Es decir, es el *interfaz* del usuario de toda la instalación de fontanería que lleva a cabo realmente la labor de quitar de enmedio lo que uno deposita en las instalaciones sanitarias. Pero por debajo de la loza y pegado a ella, están las cañerías y toda la instalación de fontanería.

Los comandos de `git` se dividen en **dos tipos**: *fontanería* o *cañería*, que son comandos que *generalmente* no ve el usuario y *loza*, que son los que ve y los que usa. Sin embargo, este capítulo trata realmente de esa fontanería, porque van a ser una serie de órdenes que se van a llevar a cabo *después* de que se ejecuten las órdenes de *loza*, o, quizás *dentro* de esas órdenes de loza.

Pero antes de usar esas órdenes de fontanería tenemos que entender cómo son las cañerías. Una parte se ha visto anteriormente: el *index* o índice que contiene todos los objetos a los que `git` debe prestarles atención a la hora de hacer un commit. Pero existen además **los objetos y las referencias**.

Los *objetos* son, en general, información que está almacenada en el repositorio. Incluyen, por supuesto, los ficheros que almacenamos en el mismo, pero también los mensajes de commit, las etiquetas y los *árboles*. Los ficheros almacenados están *divididos*: el *contenido* del fichero se almacena en un *blob* y el nombre del fichero se almacena en el árbol. Hay, pues, cuatro tipos de objetos: *blob*, *tree*, *commit* y *tag*.

La orden `ls-tree` nos permite ver qué tipos de objetos tenemos almacenados y sus códigos SHA1, que son los nombres de ficheros calculados a partir del contenido del mismo. Aunque todos están almacenados en el directorio `.git/objects`, esta orden nos permite ver también de qué tipo son:

```
~/repos-git/repo-ejemplo<master>$ git ls-tree HEAD
100644 blob a6f69e4284566cd84272c6a4e4996f64643afbea  .aspell.es.pws
100644 blob a72b52ebe897796e4a289cf95ff6270e04637aad  .gitignore
100644 blob cc5411b5557f43c7ba2f37ad31f8dc34ccda075   .gitmodules
100644 blob 4e7b6c1b5a6cb3a962ea05874d10c943c1923f39  .travis.yml
100644 blob d5445e7ac8422305d107420de4ab8e1ee6227cca  LICENSE
100644 blob d1913ebe4d9e457be617ee0e786fc8c30a237902  Procfile
```

```

100644 blob c5badda0c484c989e958ea4e27dfe11d69f3c8ef    README.md
160000 commit fa8b7521968bddf235285347775b21dd121b5c11  curso
100644 blob f8c35adaf57066d4329737c8f6ec7ce6179cc221  package.json
100644 blob 08827778af94ea4c0ddbc28194ded3081e7b0f87  shippable.yml
040000 tree 39da6b155c821af1e6a304daca9b66efb1ac651f    test
100644 blob 94f151d9ef9340c81989b0c3fa8c517c068e1864  web.js

```

En este caso tenemos objetos de tres tipos: blob, commit y tree. a `ls-tree` se le pasa un *tree-ish*, es decir, algo que apunte a dónde esté almacenado un árbol pero, para no preocuparnos de qué se trata esto, usaremos simplemente `HEAD`, que apunta como sabéis a la punta de la rama en la que nos encontramos ahora mismo. También nos da el SHA1 de 40 caracteres que representa cada uno de los ficheros. Si queremos que se expandan los `tree` para mostrar los ficheros que hay dentro también usamos la opción `-r`

```

~/repos-git/repo-ejemplo<master>$ git ls-tree -r HEAD
100644 blob a6f69e4284566cd84272c6a4e4996f64643afbea    .aspell.es.pws
100644 blob a72b52ebe897796e4a289cf95ff6270e04637aad    .gitignore
100644 blob cc5411b5557f43c7ba2f37ad31f8dc34ccda075     .gitmodules
100644 blob 4e7b6c1b5a6cb3a962ea05874d10c943c1923f39    .travis.yml
100644 blob d5445e7ac8422305d107420de4ab8e1ee6227cca    LICENSE
100644 blob d1913ebe4d9e457be617ee0e786fc8c30a237902    Procfile
100644 blob da5b5121adb42e990b9e990c3edb962ef99cb76a    README.md
160000 commit fa8b7521968bddf235285347775b21dd121b5c11  curso
100644 blob f8c35adaf57066d4329737c8f6ec7ce6179cc221  package.json
100644 blob 08827778af94ea4c0ddbc28194ded3081e7b0f87  shippable.yml
100644 blob 9920d80438d42e3b0a6924a0fcace2d53a6af602  test/route.js
100644 blob 36cc059186e7cb247eaf7bfd6a318be6cfff9ea3  views/layout.jade
100644 blob 97c32024cda29e0fb6abebf48d3f6740f0acb9e2  web.js

```

que muestra solo los objetos de tipo blob (y un commit) con el camino completo que llega hasta ellos.

Si editamos un fichero tal como el `README.md`, tras hacer el commit tendrá esta apariencia:

```

~/repos-git/repo-ejemplo<master>$ git ls-tree HEAD
100644 blob a6f69e4284566cd84272c6a4e499
[...]
100644 blob da5b5121adb42e990b9e990c3edb962ef99cb76a    README.md

```

Como vemos, ha cambiado el SHA1. Pero `ls-tree` va más allá y te puede mostrar también cuál es el estado del repositorio hace varios commits. Por ejemplo, podemos usar `HEAD^` para referirnos al commit anterior y `git ls-tree HEAD^` nos devolvería exactamente el mismo estado en el que estaba antes de hacer la modificación a `README.md`. De hecho, podemos usar también la

abreviatura del commit de esta forma `git ls-tree 5be23bb`, siendo este último una parte del SHA1 (o hash) del último commit; nos devolvería el último resultado.

Pero podemos ir todavía más profundamente dentro de las tuberías. `ls-tree` sólo lista los objetos que ya forman parte del árbol, del principal o de alguno de los secundarios. Puede que necesitemos acceder a aquellos objetos que se han añadido al índice, pero todavía no han pasado a ningún árbol. Para eso usamos `ls-files`. Tras añadir un fichero que está en un subdirectorio `views` con `add`, podemos hacer:

```
git ls-files --stage
100644 a6f69e4284566cd84272c6a4e4996f64643afbea 0    .aspell.es.pws
100644 a72b52ebe897796e4a289cf95ff6270e04637aad 0    .gitignore
100644 cc5411b5557f43c7ba2f37ad31f8dc34ccda075 0    .gitmodules
100644 4e7b6c1b5a6cb3a962ea05874d10c943c1923f39 0    .travis.yml
100644 d5445e7ac8422305d107420de4ab8e1ee6227cca 0    LICENSE
100644 d1913ebe4d9e457be617ee0e786fc8c30a237902 0    Procfile
100644 da5b5121adb42e990b9e990c3edb962ef99cb76a 0    README.md
160000 fa8b7521968bddf235285347775b21dd121b5c11 0    curso
100644 f8c35adaf57066d4329737c8f6ec7ce6179cc221 0    package.json
100644 08827778af94ea4c0ddbc28194ded3081e7b0f87 0    shippable.yml
100644 9920d80438d42e3b0a6924a0fcace2d53a6af602 0    test/route.js
100644 36cc059186e7cb247eaf7bfd6a318be6cffb9ea3 0    views/layout.jade
100644 94f151d9ef9340c81989b0c3fa8c517c068e1864 0    web.js
```

Que nos devuelve, en penúltimo lugar, un fichero que todavía no ha pasado al árbol. Evidentemente, tras el commit:

```
~/repos-git/repo-ejemplo<master>$ git ls-tree HEAD
[...]
040000 tree fd3846c0d6089437598004131184c61aea2b6514    views
```

Este listado nos muestra el nuevo objeto de tipo `tree` que se ha creado y nos da su SHA1, que podemos usar para examinarlo con `ls-tree`

```
~/repos-git/repo-ejemplo<master>$ git ls-tree fd3846c
100644 blob 36cc059186e7cb247eaf7bfd6a318be6cffb9ea3    layout.jade
```

que, si queremos ver en una vista más normal, hacemos lo mismo con `ls-file`

```
~/repos-git/repo-ejemplo<master>$ git ls-files views
views/layout.jade
```

Hay un tercer comando relacionado con el examen de directorios y ficheros locales, `cat-file`, que muestra el contenido de un objeto, en general. Por ejemplo, en este caso, para listar el contenido de un objeto de tipo `tree`

```
~/repos-git/repo-ejemplo<master>$ git cat-file -p fd3846c
100644 blob 36cc059186e7cb247eaf7bfd6a318be6cffb9ea3    layout.jade
```

nos muestra que ese objeto contiene un solo fichero, `layout.jade`, y sus características. Pero más curioso aún es cuando se usa sobre objetos de tipo `commit` (no sobre el objeto `commit` que aparece arriba, que se trata de un `commit` de otro repositorio al contener el directorio `curso` un submódulo de git. Por ejemplo, podemos hacer:

```
~/repos-git/repo-ejemplo<master>$ git cat-file -p HEAD
tree 1c40899a32c2b5ec7f930bd943e5d5bb98562d373
parent 5be23bb2a610260da013fcea807be872a4bd6981
author JJ Merelo <jjmerelo@gmail.com> 1397752151 +0200
committer JJ Merelo <jjmerelo@gmail.com> 1397752151 +0200
Añade layout
```

que, dado que `HEAD` apunta al último commit, nos muestra en modo *pretty-print* toda la información sobre el último `commit` y muestra el árbol de ficheros correspondiente, que podemos listar con

```
~/repos-git/repo-ejemplo<master>$ git cat-file -p 1c40899a
100644 blob a6f69e4284566cd84272c6a4e4996f64643afbea    .aspell.es.pws
100644 blob a72b52ebe897796e4a289cf95ff6270e04637aad    .gitignore
100644 blob cc5411b5557f43c7ba2f37ad31f8dc34ccda075     .gitmodules
100644 blob 4e7b6c1b5a6cb3a962ea05874d10c943c1923f39    .travis.yml
100644 blob d5445e7ac8422305d107420de4ab8e1ee6227cca    LICENSE
100644 blob d1913ebe4d9e457be617ee0e786fc8c30a237902    Procfile
100644 blob da5b5121adb42e990b9e990c3edb962ef99cb76a    README.md
160000 commit fa8b7521968bddf235285347775b21dd121b5c11  curso
100644 blob f8c35adaf57066d4329737c8f6ec7ce6179cc221    package.json
100644 blob 08827778af94ea4c0ddbc28194ded3081e7b0f87    shippable.yml
040000 tree 39da6b155c821af1e6a304daca9b66efb1ac651f    test
040000 tree fd3846c0d6089437598004131184c61aea2b6514    views
100644 blob 97c32024cda29e0fb6abebf48d3f6740f0acb9e2    web.js
```

En general, si queremos ahondar en las entrañas de un punto determinado en la historia del repositorio, trabajar con `ls-files`, `cat-file` y `ls-tree` permite obtener toda la información contenida en el mismo. Esto nos va a resultar útil un poco más adelante.

**Viva la diferencia** En muchos casos para procesar los cambios dentro de un gancho necesitaremos saber cuál es la diferencia con versiones anteriores del fichero. Hay que tener en cuenta que esas diferencias, dependiendo del estado en el que estemos, estarán en el árbol o en el índice preparadas para ser enviadas al repositorio. En general, son una serie de órdenes con **diff** en ellas. La más simple, **git diff**, nos mostrará la diferencia entre los archivos en el índice y el último commit.

```
~/repos-git/repo-ejemplo<master>$ git diff
diff --git a/views/layout.jade b/views/layout.jade
index 36cc059..2a66d58 100644
--- a/views/layout.jade
+++ b/views/layout.jade
@@ -1,6 +1,11 @@
-!!! 5
+doctype html
+html(lang="en")

-body
+html
+  head
+    title #{title}
+  body

-#wrapper
-  block content
\ No newline at end of file
+  h1 Curso de git
+
+  block content
\ No newline at end of file
diff --git a/web.js b/web.js
index 97c3202..93b6255 100644
--- a/web.js
+++ b/web.js
@@ -27,6 +27,12 @@ app.get('/', function(req, res) {
    res.send(routes['README']);
  });

+app.get('/curso/:ruta', function(req, res) {
+  var ruta = "curso/texto/"+req.params.ruta;
+  console.log("Request "+req.params.ruta + " doc " + ruta + " contenido " + file_contenido);
+  res.render('doc', { content: routes[ruta], title: ruta });
+})
+
+app.get('/curso/texto/:ruta', function(req, res) {
```

```
//          console.log("Request "+req.params.ruta);
          var ruta_toda = "curso/texto/"+req.params.ruta;
```

Esta vista de [diff](#) las diferencias sigue el formato habitual en [la utilidad diff](#), que permite generar parches para aplicarlos a conjuntos de ficheros. En concreto, muestra qué ficheros se están comparando (pueden ser diferentes ficheros, si se ha cambiado el nombre) los SHA1 de los contenidos correspondientes, y luego un + o - delante de cada una de las líneas que hay de diferencia. Este fichero se podría usar directamente con la utilidad `diff` de Linux, pero realmente no nos va a ser de mucha utilidad a la hora de saber, por ejemplo, qué ficheros se han modificado. Para hacer esto, [simplemente](#):

```
~/repos-git/repo-ejemplo<master>$ git diff --name-only
views/layout.jade
web.js
```

que se puede hacer un poco más completa con `--name-status`:

```
~/repos-git/repo-ejemplo<master>$ git diff --name-status
M      views/layout.jade
M      web.js
~/repos-git/repo-ejemplo<master>$ git diff --name-status --cached
A      views/doc.jade
```

que nos muestra, usando una sola letra, qué tipo de cambio han sufrido. En el primer caso nos muestra que han sido Modificados, y en el segundo caso, además usamos otra opción, `--cached` que, en este caso, nos muestra los ficheros que han sido preparados para el commit; es decir, la [diferencia que hay entre la cabeza y el índice](#); en este caso, “A” indica que se trata de un fichero añadido. Podemos ver todo junto con

```
~/repos-git/repo-ejemplo<master>$ git diff --name-status HEAD
A      views/doc.jade
M      views/layout.jade
M      web.js
```

que nos muestra la diferencia entre el commit más moderno (HEAD) y el área de trabajo ahora mismo: hemos cambiado dos ficheros y añadido uno. Un resultado similar obtendremos con `diff-index`, cuya principal diferencia es que compara siempre el índice con algún *árbol*, sin tener ningún valor por omisión:

```
~/repos-git/repo-ejemplo<master>$ git diff-index HEAD
:000000 100644 0000000000000000000000000000000000 67e6d7e1ecbb64ff7d467dc2103fa2b2f
:100644 100644 36cc059186e7cb247eaf7bfd6a318be6cffb9ea3 00000000000000000000000000000000
:100644 100644 97c32024cda29e0fb6abebf48d3f6740f0acb9e2 00000000000000000000000000000000
```

Además del estado muestra el hash inicial y final de cada uno de los ficheros. En este caso, como todavía no le hemos hecho commit, muestra 0. **diff-tree**, sin embargo, sí muestra el SHA1 puesto que trabaja con el árbol:

```
~/repos-git/repo-ejemplo<master>$ git diff-tree HEAD
fe88e5eefff7f3b7ea95be510c6dc87054bbcb0
:000000 040000 000000000000000000000000000000000000000000000000 fd3846c0d6089437598004131184c61ae
:100644 100644 94f151d9ef9340c81989b0c3fa8c517c068e1864 97c32024cda29e0fb6abebf48d3f6740f
```

Aunque en este caso muestra un árbol, **views**, que ha sido cambiado porque se le ha añadido un fichero nuevo, **views/doc.jade**. En el momento que se haga el commit y pase por tanto del índice al la zona de *staging*, los hash ya están calculados y cambia la salida:

```
~/repos-git/repo-ejemplo<master>$ git diff-tree HEAD
637c2820013188f1c4951aef0c21de20440a6fbb
:040000 040000 fd3846c0d6089437598004131184c61aea2b6514 6bb4560a218c008bbc468f23f36f26ff6:
:100644 100644 97c32024cda29e0fb6abebf48d3f6740f0acb9e2 93b625533c2d1752d9a8e789878512919:
```

diff-index, sin embargo, no devolverá nada puesto que todos los cambios que se habían hecho han pasado al árbol.

En general, lo que más nos va a interesar a la hora de hacer un gancho es qué ficheros han cambiado. Pero conviene conocer toda la gama de posibilidades que ofrece **git**, sobre todo para poder entender su estructura interna.

**Los dueños de las tuberías** No todo el contenido que hay en el repositorio son los ficheros que forman parte del mismo. Hay una parte importante de la fontanería que son los metadatos del repositorio. Hay dos órdenes importantes, **var** y **config**. Con **-l** nos listan todas las variables o variables de configuración disponibles

```
~/repos-git/curso-git/texto<master>$ git var -l
user.email=jjmerelo@gmail.com
user.name=JJ Merelo
filter.obj-add.smudge=cat
push.default=simple
rerere.enabled=true
core.repositoryformatversion=0
core.filemode=true
core.bare=false
core.logallrefupdates=true
remote.origin.url=git@github.com:oslugr/curso-git.git
remote.origin.fetch=+refs/heads/*:refs/remotes/origin/*
```

```
branch.master.remote=origin
branch.master.merge=refs/heads/master
GIT_COMMITTER_IDENT=JJ Merelo <jjmerelo@gmail.com> 1399019391 +0200
GIT_AUTHOR_IDENT=JJ Merelo <jjmerelo@gmail.com> 1399019391 +0200
GIT_EDITOR=editor
GIT_PAGER=pager
```

Todas excepto las cuatro últimas variables son variables de configuración que, por tanto, se pueden obtener también con `git config -l`. Por sí sólo, `config` o `var` listan el valor de una variable:

```
~/repos-git/curso-git/texto<master>$ git config user.name
JJ Merelo
```

La mayoría de estos valores están disponibles o como variables de entorno o en ficheros; sin embargo estas órdenes dan un interfaz común para todos los sistemas operativos.

Todavía nos hacen falta una serie de órdenes para tomar decisiones sobre ficheros y sobre dónde estamos en el repositorio. La veremos a continuación

**Simplemente, `rev-parse`** La [tersa descripción del comando `rev-parse`](#), “[recoge y procesa parámetros](#)” esconde la complejidad del mismo y su potencia, que va desde el procesamiento de parámetros hasta la especificación de objetos, pasando por la búsqueda de diferentes directorios dentro del repositorio git. Por ejemplo, se puede usar para verificar si un objeto existe o no:

```
~/repos-git/repo-ejemplo<master>$ git rev-parse --verify HEAD
637c2820013188f1c4951aef0c21de20440a6fbb
```

Nos muestra el SHA1 de la cabeza actual del repositorio de ejemplo, verificando que actualmente existe. No lo hará si acabamos de crear el repositorio, por ejemplo

```
/tmp/pepe<>$ git init
Initialized empty Git repository in /tmp/pepe/.git/
/tmp/pepe<>$ git rev-parse --verify HEAD
fatal: Needed a single revision
```

De hecho, con él podemos encontrar todo tipo de objetos usando la notación que permite especificar revisiones

```
~/repos-git/repo-ejemplo<master>$ git rev-parse HEAD@{1.month}
61253ecba351921c96a1553f6c5b7f9910f286f3
```



que correspondería a [este commit del 9 de marzo](#) y que podemos listar usando `show`, `describe` o cualquiera de los otros comandos que se pueden aplicar a objetos. En general, para esto se usará dentro de los *garfios*: para poder acceder a un objeto determinado o a sus metadatos a la hora de ver las diferencias con el objeto actual.

## Concepto de *hooks*

Un *hook* literalmente *garfio* o *gancho* es un programa que se ejecuta cuando sucede un evento determinado en el repositorio. Los *webhooks* de GitHub, por ejemplo, son un ejemplo: cuando se lleva a cabo un *push*, se envía información al sitio configurado para que ejecute un programa determinado: pase unos tests, publique un tweet, o lleve a cabo una serie de comprobaciones.

Los *ganchos* no son estrictamente necesarios en todo tipo de instalaciones; se puede trabajar con un repositorio sin tener la necesidad de usarlos. Sin embargo, [son tremendamente útiles para automatizar una serie de tareas](#) (como los tests que se usan en integración continua), implementar una serie de políticas para todos los usuarios de un repositorio (formato de los mensajes de *commit*, por ejemplo) y añadir información al repositorio de forma automática.

Los *hooks* son, por tanto, programas ejecutables. Cualquier programa que se pueda lanzar puede servir, pero generalmente se usa o guiones del *shell* (si uno es suficientemente masoquista) o lenguajes de *scripting* tales como Perl, Python, Ruby, Javascript o, si uno es *realmente* masoquista, PHP. En realidad a git le da igual qué lenguaje se use.

Los *hooks* van en su propio directorio, `.git/hooks` que se crea automáticamente y que tiene, siempre, una serie de *scripts* ejemplo, ninguno de ellos activados. Sólo se admite un *hook* por evento, y ese *hook* tendrá el nombre del evento asociado; es decir, un programa llamado `post-merge` en ese directorio se ejecutará siempre cada vez que se termine un *merge* con éxito. Como generalmente uno quiere que los scripts tengan un nombre razonable, la estrategia más general es usar un *enlace simbólico* de esta forma

```
ln -s nombre-real-del-script.sh post-merge
```

y, en todo caso, no se debe olvidar

```
chmod +x nombre-real-del-script.sh
```

para hacerlo ejecutable.

Windows seguramente tendrá su forma particular de hacer lo mismo, o ninguna. Por cualquiera de esas dos razones, nunca recomendamos Windows como una plataforma para desarrollo. Úsala para

la declaración de la renta o para jugar al Unreal, pero para desarrollar usa una plataforma para desarrolladores: Linux (o Mac, que tiene un núcleo Unix por debajo).

Los *hooks* se activarán cuando se ejecute un comando determinado y recibirán una serie de parámetros como argumento o en algún caso como entrada estándar. Este [cuadro](#) resume cuando se ejecutan y también qué reciben como parámetro. En general, también tendrán influencia en si tiene éxito o no el comando determinado: salir con un valor no nulo, en algunos casos, parará la ejecución del comando con un mensaje de error. Por ejemplo, un *hook applypatch-msg*, que se aplica desde el comando `git am` antes de que se ejecute, parará la aplicación del parche si se sale con un valor 1.

De todos los *hooks* posibles sólo veremos los que se refieren al *commit*. Son los que se pueden usar en local (los referidos a *push* sólo se programan en remoto, y los que se aplican a `git am` o `git gc` quedan fuera de los temas de este libro. Hay sólo cuatro de estos, que veremos a continuación.

### Programando un *hook* básico

En general, un *hook* hará lo siguiente

1. Examinar el entorno y los parámetros de entrada
2. Hacer cambios en el entorno, los ficheros o la salida
3. Salir con un mensaje si hay algún error, o ninguno.

No son diferentes de ningún otro programa, en realidad, salvo que los parámetros de entrada y cómo se debe salir está establecido de antemano.

Miremos un script simple, que actúa como gancho para preparación de un mensaje de commit (`prepare-commit-msg`, incluido en el [repositorio de ejemplo de este curso](#))

```
#!/bin/sh
SOB=$(git config github.user)
grep -qs "^$SOB" "$1" || echo ". Cambio por @$SOB" >> "$1"
```

Este script tiene toda la simplicidad de estar en dos líneas y toda la complicación de estar escrito para el *shell*. [Esta introducción](#) venerable te puede ayudar a empezar a trabajar con él, pero en este capítulo no pretendemos que aprendas a programar, sólo que tengas las nociones básicas para echar a andar y posiblemente modificar ligeramente un gancho.

Empecemos por la primera línea: es común a todos los guiones del shell. Simplemente indica el camino en el que se encuentra el mismo, con `#!` indicando

que se trata de un fichero ejecutable (junto con el `chmod +x`, que se lo indica al sistema de ficheros).

La siguiente línea define una variable, SOB, que no es acrónimo de nada, cuidado. Esa variable usa el formato de ejecución de un comando del shell `$()` para asignar la salida de dicho comando a la variable.

La tercera línea, en resumen, comprueba si el nombre del usuario ya está en el mensaje y si no lo está le añade una “firma” que lo incluye. Lo hace de la forma siguiente: `grep -qs "^$SOB" "$1"` comprueba si el valor de la variable no (de ahí el caret `^`) está ya en el mensaje (cuyo nombre de fichero está en el primer argumento, `$1`, según vemos en la [chuleta de ganchos de git](#). Si no lo está (de ahí el `||`, es decir, *O*, se escribe (`echo`) el valor de la variable añadiéndose (`>>`) al final del fichero cuyo nombre está en la variable `$1`.

La variable `github.user` no tiene por qué estar definida siempre. Se puede sustituir por `user.email`, por ejemplo, o por `user.name`, que sí se suele definir siempre cuando se crea un repositorio.

Este [otro ejemplo](#) es todavía más minimalista, y también añade información al commit (que, como en el caso anterior, se puede eliminar si se edita el fichero de commit):

```
STATS=$(git diff --cached --shortstat)
echo ". Cambios en este commit\n ${STATS}" >> "$1"
```

Es interesante notar, en este caso, que se usa `diff --cached` ya que en este caso los cambios estarán ya *staged* o *cached* y la diferencias (que suelen aparecer de todas formas en el mensaje que da el comando) serán entre HEAD y lo que hay ya almacenado el área de preparación de los ficheros, no entre HEAD y lo que hay en el sistema de ficheros; esto es así porque ya estamos *dentro* del commit y los ficheros están ya preparados para ser procesados en las tuberías de `git` por sus comandos-tubería. En resumen, esta orden da una mini-estadística que dice el número de ficheros y líneas cambiadas, produciendo [resultados sobre este mismo fichero tales como este](#):

Mini-corrección

```
. Cambios en este commit
 1 file changed, 1 insertion(+)
```

Otro *hook* puede servir para comprobar que los mensajes de *commit* son correctos. Como se ha visto anteriormente, una buena práctica es usar una primera línea de 50 caracteres (que aparecerán como título) seguida por una línea vacía y el resto del mensaje. Esto se puede aplicar mediante [un programa en Python](#) o el siguiente en [Node](#), una versión de Javascript.

```
#!/usr/bin/env node

var fs = require('fs');
var msg_file = process.argv[2];

fs.readFile( msg_file, 'utf8', function( err, data ) {
    if ( err ) throw err;
    var lines = data.split("\n");
    if ( lines[0].length > 50 ) {
        console.log("[FORMATO] Primera línea > 50 caracteres");
        process.exit(1);
    }
})
```

La primera línea es común a los scripts, y a continuación se carga el módulo necesario para usar el sistema de ficheros (**fs**) y se lee el argumento que se pasa por línea de órdenes, el nombre del fichero que contiene el mensaje del commit (este mensaje estará ahí aunque lo hayamos pasado con **-m** desde la línea de órdenes).

El resto, usando el modo asíncrono que es común en Node, lee el fichero (creando una excepción si hay algún error), lo divide en líneas usando **split** y a continuación comprueba si la primera línea (**lines[0]**) tiene más de 50 caracteres, en cuyo caso sale del proceso con un código de error (1), de esta forma

```
~/repos-git/curso-git<master>$ git commit -am "Añadiendo un montón de cosas al capítulo de 1
[FORMATO] Primera línea > 50 caracteres
```

Si no es así, simplemente deja pasar el mensaje. En este *hook*, curiosamente, no se usa más comando de git que el mensaje almacenado en el fichero; realmente, no es imprescindible usarlo, sólo cuando vayamos a usar información de git en el mismo.

El programa anterior es un ejemplo de cómo se pueden implementar políticas de formato o de cualquier otro tipo sobre un repositorio; sin embargo, la capacidad que tienen es limitada, ya que se aplican sólo sobre los mensajes. En realidad, el que actúen de esa forma es convencional, porque los programas que ejecutan los *ganchos* se diferencian solamente en el momento en el que actúan, no en lo que pueden hacer. Sin embargo, si queremos [implementar una política](#) sobre los nombres de ficheros o el contenido de los mismos, [hay que usar un gancho que actúe cuando se añadan al repositorio](#) como [el siguiente gancho pre-commit](#) escrito en Perl:

```
#!/usr/bin/env perl
my $is_head = `git rev-parse --verify HEAD`;
my $last_commit = $is_head?"HEAD":"4b825dc642cb6eb9a060e54bf8d69288fbee4904";
```

```

my @changes = `git diff-index --name-only $last_commit`;
my %policies = ( no_underscore => qr/_/ );
for my $f (@changes) {
    for my $p ( keys %policies ) {
        if ($f =~ /$policies{$p}/) {
            die "[FORMATO]: $p ";
        }
    }
}
}

```

Este programa es similar a los anteriores, pero para empezar usa un comando *cañería* que se encuentra mucho: `rev-parse`. Las dos primeras órdenes comprueban si nos encontramos en el primer *commit* de un repositorio (en cuyo caso `HEAD` no existe y tendremos que usar el *commit primigenio* que es igual en todos los repositorios de `git`; lo que pretenden esas dos líneas es encontrar el “último commit” para ver cuál es la diferencia con respecto a él. Como vamos a tratar con ficheros que acaban de ser añadidos al índice, usamos `diff-index` en la siguiente línea con un formato que nos devolverá solamente los ficheros cambiados desde el último commit (o desde el primero) sin que nos interese nada más sobre ellos.

La siguiente línea define un *hash* con un nombre y una expresión regular que será la que se tiene que implementar. Si no conoces el lenguaje no te preocupes, pero si lo conoces (ese u otro) es relativamente fácil añadir políticas nuevas, como por ejemplo que no se permitan `.pdfs`, simplemente añadiéndole una línea.

El bucle `for` posterior es que va recorriendo cada uno de los cambios y cada una de las políticas (aunque en este caso habrá una sola) y saldrá con `die` si alguno de los ficheros tiene un nombre incorrecto.

```

~/repos-git/repo-plantilla<master>$ git commit -am "A ver si me deja"
[FORMATO]: no_underscore at .git/hooks/pre-commit line 13.
~/repos-git/repo-plantilla<master>$ git status
# En la rama master
# Cambios para hacer commit:
#   (use «git reset HEAD <archivo>...«para eliminar stage)
#
#   archivo nuevo:   uno_que_no

```

Lo que se puede hacer ahora no es tan trivial: puedes cambiar el fichero de nombre a pelo, pero ya está en el índice, por eso es mejor hacer algo como `git mv` (o `git rm --force`).

### Algunos *hooks* útiles explicados

Hay múltiples posts en blogs que [explican diferentes ejemplos de hooks](#) y lo que se puede hacer con ellos. Por ejemplo, en [este conjunto](#) usa programas externos y su

código de salida (\$?) para comprobar si un programa es correcto o no; también usa extensivamente `git stash` para almacenar todos los ficheros que se hayan modificado y luego los recupera con `git stash pop`. [Este, por ejemplo](#), crea una plantilla para usarla en los mensajes de commit, Pero [este gist](#) incluye algunos *hooks* útiles que vamos a ver. Por ejemplo, [el siguiente](#) comprueba si se encuentra la palabra `debugger` en el fichero (es decir, comentarios de depuración):

```
if git-rev-parse --verify HEAD >/dev/null 2>&1; then
    against=HEAD
else
    against=4b825dc642cb6eb9a060e54bf8d69288fbee4904
fi
for FILE in `git diff-index --check --name-status $against -- | cut -c3-` ; do
# Check if the file contains 'debugger'
if [ "grep 'debugger' $FILE" ]
then
echo $FILE ' contains debugger!'
exit 1
fi
done
```

Lo más complicado que tiene este fichero es el uso de filtros del shell (algo que, por otro lado, debería aprenderse); esos filtros hacen algo similar a lo que se ha hecho antes con Perl: recorre el nombre de los ficheros y le aplica el buscador de cadenas, `grep`, buscando la palabra `debugger`; si la encuentra, sale. Quizás está mejor explicado en [la historia original](#)

## Contents

<b>Aprende git</b>	<b>1</b>
<b>Pablo Hinojosa y JJ Merelo</b>	<b>1</b>
Licencia . . . . .	1
Prólogo y agradecimientos . . . . .	1
Agradecimientos adicionales . . . . .	1
Introducción . . . . .	2
Objetivos . . . . .	2
Software Libre . . . . .	2
Sistemas de control de Versiones . . . . .	2

Tipos de Sistemas de Control de Versiones . . . . .	3
git . . . . .	4
Abrir una cuenta en Github . . . . .	5
Uso básico de git . . . . .	5
Objetivos . . . . .	5
Instalar git . . . . .	6
Clientes GUI para Linux, Windows y Mac . . . . .	15
Empezando a usar git . . . . .	15
¿Cómo funciona git? . . . . .	18
Manteniendo nuestro repositorio al día . . . . .	18
Sincronizando repositorios . . . . .	21
Comportamiento por defecto de push . . . . .	24
El archivo .gitignore . . . . .	25
Solución de problemas con git . . . . .	27
Objetivos . . . . .	27
Obtener Ayuda . . . . .	27
Viendo el historial . . . . .	27
Borrado de archivos . . . . .	28
Rehacer un commit . . . . .	29
Deshacer cambios en un archivo . . . . .	29
Resolviendo conflictos . . . . .	29
Retrocediendo al pasado . . . . .	30
Viendo (y recuperando) archivos antiguos . . . . .	31
Más usos de git . . . . .	31
Objetivos . . . . .	31
Flujos de desarrollo de software (y quizás de otras cosas) . . . . .	31
Organización de un repositorio de git . . . . .	32
Flujos de trabajo con git . . . . .	35
Ramas . . . . .	37
Los misterios del rebase . . . . .	44
Quién hizo qué . . . . .	45

Usando <b>git</b> como los profesionales: GitHub . . . . .	47
Objetivos . . . . .	47
Por qué GitHub . . . . .	47
Repositorios públicos y privados . . . . .	48
El GitHub <i>social</i> . . . . .	48
Creando páginas para GitHub pages . . . . .	50
Cómo usar los hooks . . . . .	51
Algunos <i>hooks</i> interesantes: sistemas de integración continua . . .	52
Cliente de GitHub . . . . .	54
Listo para el lanzamiento: publicación en GitHub . . . . .	55
<i>Hooks</i> : ejecutando código tras una orden git . . . . .	56
Objetivos . . . . .	56
Viendo las cañerías: estructura de un repositorio <b>git</b> . . . . .	56
Paso a paso . . . . .	59
El nombre de las cosas: refiriéndonos a objetos en git. . . . .	59
Comandos de alto y bajo nivel: <i>fontanería</i> y <i>loza</i> . . . . .	62
Concepto de <i>hooks</i> . . . . .	70
Programando un <i>hook</i> básico . . . . .	71
Algunos <i>hooks</i> útiles explicados . . . . .	74